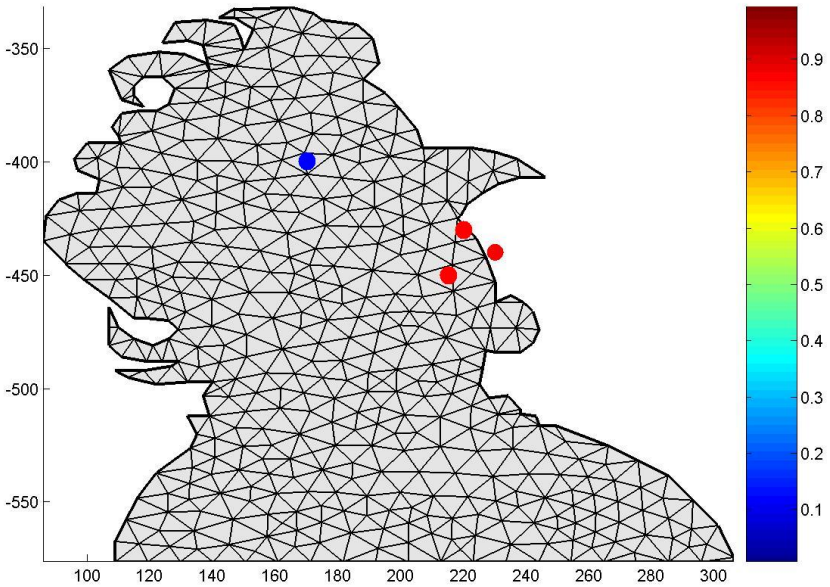


Geostatistical analysis tools

Diva User's Guide

(last modified February 28, 2007)

J.-M. Beckers[1] (JM.Beckers@ulg.ac.be),
D. Sirjacobs[2] (D.Sirjacobs@ulg.ac.be),
C. Troupin[2] (ctroupin@ulg.ac.be)

GeoHydrodynamics and Environment Research, MARE,
University of Liège, Sart-Tilman B5, 4000 Liège, BELGIUM

[1] Honorary Research Associate, National Fund for Scientific Research, BELGIUM
[2] PhD Student, Fonds pour la formation à la Recherche dans l'Industrie et dans l'Agriculture (F.R.I.A), BELGIUM

Contents

I **5**

1 Diva theory **6**

1.1 The gridding problem 6

1.1.1 Interpolation vs. approximation 7

1.1.2 Data analysis and gridding methodologies 7

1.1.3 Noise on data 8

1.2 VIM and its implementation Diva 8

1.2.1 Formulation 9

1.2.2 Error field 11

1.2.3 Kernel and correlation function 11

1.2.4 Domain gridding 12

1.3 Comparison between the methods 13

1.4 Additional tools 14

1.4.1 Generalized Cross Validation (GCV) 14

1.4.2 Advection constraint 14

1.4.3 DINEOF 14

1.4.4 Possible developments 14

2 Installation procedure **16**

2.1 Tcl/Tk 16

2.1.1 Windows 16

2.1.2 Linux 18

2.2 Diva software installation 18

2.2.1 Diva-4.1 content 19

2.2.2 Command Line version . . . 19
2.2.3 Graphical User Interface version . . . 19
2.3 Compiling . . . 21
2.3.1 Tcl/Tk programs . . . 22
2.4 Creation of shortcuts . . . 22
2.4.1 Windows . . . 22
2.4.2 Linux . . . 23
2.5 PlPlot library . . . 23

3 Diva Use **24**
3.1 General . . . 24
3.1.1 Data . . . 24
3.1.2 Contour . . . 25
3.1.3 Finite Element mesh . . . 25
3.1.4 Parameters . . . 26
3.2 Command Line version . . . 28
3.2.1 Working directories . . . 28
3.2.2 Input files . . . 28
3.2.3 Output files . . . 29
3.2.4 Running an analysis . . . 29
3.3 Graphical Users Interface . . . 31
3.3.1 Working directory . . . 31
3.3.2 The `Data` menu . . . 32
3.3.3 The `mesh` menu . . . 34
3.3.4 The `Analysis` menu . . . 36
3.4 The `Options` menu . . . 37
3.5 Generation of contour and topography files . . . 38
3.5.1 Coastline generation . . . 38
3.5.2 Topography generation . . . 40
3.6 Display of outputs . . . 41
3.6.1 Matlab . . . 41
3.6.2 NcBrowse and Ncview . . . 42

4 Test cases **44**

4.1 Island . . . 44

4.1.1 Data . . . 44

4.1.2 Mesh . . . 45

4.1.3 Analysis . . . 45

4.2 Peninsula . . . 46

4.2.1 Data . . . 46

4.2.2 Mesh . . . 47

4.2.3 Analysis . . . 47

4.3 Interior sea . . . 48

4.3.1 Data . . . 48

4.3.2 Mesh . . . 48

4.3.3 Analysis . . . 49

4.4 Noise . . . 50

4.4.1 Data . . . 50

4.4.2 Mesh . . . 50

4.4.3 Analysis . . . 51

4.5 Single point . . . 51

4.5.1 Data . . . 51

4.5.2 Mesh . . . 52

4.5.3 Analysis . . . 52

5 A complete example with the GUI **53**

5.1 Data Extraction . . . 53

5.2 Mesh generation . . . 54

5.3 Analysis . . . 55

6 Generalized Cross Validation in Diva **58**

6.1 Introduction . . . 58

6.1.1 Generalities . . . 58

6.1.2 Ordinary Cross Validation (OCV) . . . 60

6.1.3 Generalized Cross Validation (GCV) . . . 60

6.2 Quality control (QC) of data . . . 62
6.2.1 Quality criteria . . . 62
6.2.2 Normalized version . . . 63
6.3 Application and implementation in Diva . . . 63
6.3.1 Diva formulation . . . 64
6.3.2 Implementation of result validation in Diva . . . 64

II Appendix **66**

A Diva code **67**
A.1 Fortran code . . . 67
A.1.1 Diva programs . . . 67
A.1.2 Mesh programs . . . 68
A.1.3 GUI programs . . . 68
A.1.4 Utilities . . . 69
A.1.5 NetCDF output . . . 70
A.2 Input and output files . . . 70
A.3 Tcl/Tk code . . . 73

B Gallery **75**

Bibliography **80**
DINEOF . . . 80
Diva . . . 80
Miscellaneous . . . 81

Useful links **82**

Conditions of use **83**

Acknowledgements **84**

Part I

1

Diva theory

This chapter is dedicated to the description of the theorical background necessary to understand how the Diva software works. As only the main ideas are summarized, the reader is invited to refer to the articles mentioned in the bibliography.

1.1 The gridding problem

In oceanography a typical concern consists in determining a field $\varphi(\mathbf{r})$ on a regular grid of positions $\mathbf{r}$, knowing N_d data in locations $\mathbf{r_j}, j = 1, \ldots, N_d$. This is called the *gridding problem* (Fig. 1.1).

It is useful for many applications: data analysis, graphical display, forcing or initialization of a model, ...

In this chapter we consider only two-dimensional cases, but generalization can be done to 3D and even 4D (using "distance" in time, but beware of autocorrelations as in seasonal signals).

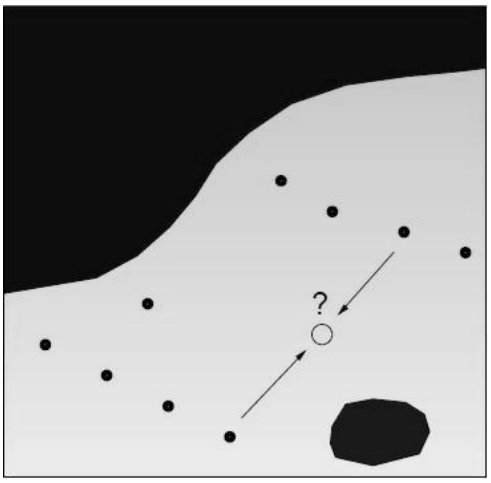


Figure 1.1: The gridding problem.

1.1.1 Interpolation vs. approximation

When treating the gridding problem, two main techniques have to be distinguished:

1. The *interpolation*, which implies a strict passing of the solution by the points of data. Physically, this means that you assume you have no error on the data.
2. The *approximation* (or *analysis*) provides a solution more smooth than the one given by the interpolation: in this case, the solution does not necessarily have to contain all the data points. The solution passes *near* the data points, so its shape is still influenced by the data. This technique allows to take into account the error on data, as well as to treat multiples data on the same location, but with different values.

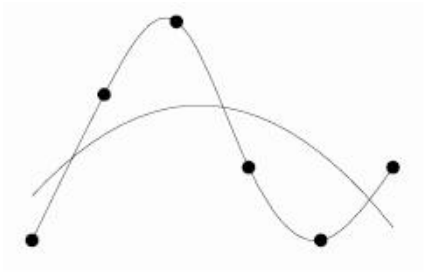


Figure 1.2: Interpolation vs. approximation.

1.1.2 Data analysis and gridding methodologies

Because of data error and often close data points, the considered field is always reconstructed with the help of approximation, never with a strict interpolation.

We have to differentiate the *subjective* analysis, for which the way the approximation is performed are decided *by hand*, and the *objective* analysis, which is based on predefined mathematical operations. The *data assimilation* uses in addition physical/biochemical dynamic governing equations.

Choice of a method: among the three previous methods, our choice will be the objective analysis. Indeed, the subjective analysis is not sufficiently... objective, and the data assimilation depends on the region and the model.

Instead of working with the data themselves, we will work with *anomalies* (φ') with respect to a *background field* (φ_b):

$$\varphi(\mathbf{r}) = \varphi_b(\mathbf{r}) + \varphi'(\mathbf{r}). \tag{1.1}$$

The background field φ_b is defined *a priori* and the anomalies are calculated with respect to this reference field (examples: climatological average, linear regression, theorical solution).

We assume that the anomalies can be expressed as a linear combination of the data, *i.e.*

$$\varphi(\mathbf{r}) = \varphi_b(\mathbf{r}) + \sum_{j=1}^{N_d} w_j \, d_j, \tag{1.2}$$

where d_j is the data anomay at the point $\mathbf{r} = \mathbf{r_j}$ and w_j is the relative weight of the data j. Now the problem consists in determining the weighting functions.

Once the background field and the weights are known, the field can be computed at any position $\mathbf{r}$, hence gridding is possible.

1.1.3 Noise on data

When measuring a field, there is always uncertainty on the value obtained (whatever the instrument and the field). *Noise* not only takes into account *instrumental* error (which is generally low), but also:

- *representativity* errors (what you measure is not what you intend to analyse);
  e.g. skin temperature, inadequate scales, ...
- *synopticity* errors (when the measures are assumed to be taken at the same time);
  e.g. data from a cruise.

Perfect fit to data is therefore not advised. In Diva, the analysis behavior is controlled via the *signal/noise ratio* of data.

1.2 VIM and its implementation Diva

VIM stands for **V**ariational **I**nverse **M**ethod. This method was initially designed for climatology: in that case, you have generally high resolution vertical profiles, sufficient profiles in all seasons, but irregular horizontal coverage [Brasseur et al. (1996)]. Thus a spatial analysis on horizontal planes is needed.

Relatively large number of data points in each plane penalises Optimal Interpolation (O.I.) methods, because these methods require the inversion of a N_d-by-N_d matrix, *i.e.* a number of operations proportional to N_d^3 (see Tab. 1.1).

This is the reason why VIM resorts to the expertise in efficient finite-element solvers.

Diva stands for **D**ifferential **I**ntegral **V**ariational **A**nalysis and is the implementation of VIM. It is designed to solve 2-D differential or variational problems of elliptic type, using the finite element method.

1.2.1 Formulation

We are looking for the field φ which minimizes the variational principle:

$$J[\varphi] = \sum_{j=1}^{Nd} \mu_j \left[d_j - \varphi(x_j, y_j)\right]^2 + \|\varphi - \varphi_b\|^2 \tag{1.3}$$

with

$$\|\varphi\| = \int_D (\alpha_2 \boldsymbol{\nabla}\boldsymbol{\nabla}\varphi : \boldsymbol{\nabla}\boldsymbol{\nabla}\varphi + \alpha_1 \boldsymbol{\nabla}\varphi \cdot \boldsymbol{\nabla}\varphi + \alpha_0 \varphi^2)\, dD \tag{1.4}$$

where

- α_0 penalizes the field itself (anomalies),
- α_1 penalizes gradients (no trends),
- α_2 penalizes variability (regularization),
- μ penalizes data-analysis misfits (objective).

Without loss of generality we can chose $\alpha_2 = 1$ (homogeneous function).

Meaning of the parameters

Writing Eq. (1.3) and (1.4) in non-dimensional form (with $\frac{1}{L}\tilde{\boldsymbol{\nabla}} = \boldsymbol{\nabla}$) , we have

$$J[\varphi] = \sum_{j=1}^{Nd} \mu[d_j - \varphi(x_j, y_j)]^2 + \int_{\tilde{D}} \left(\frac{1}{L^4} \tilde{\boldsymbol{\nabla}}\tilde{\boldsymbol{\nabla}}\varphi : \tilde{\boldsymbol{\nabla}}\tilde{\boldsymbol{\nabla}}\varphi + \frac{\alpha_1}{L^2} \tilde{\boldsymbol{\nabla}}\varphi \cdot \tilde{\boldsymbol{\nabla}}\varphi + \alpha_0 \varphi^2 \right) L^2\, d\tilde{D} \tag{1.5}$$

and multiplying by L^2:

$$J[\varphi] = \sum_{j=1}^{Nd} \mu L^2 [d_j - \varphi(x_j, y_j)]^2 + \int_{\tilde{D}} \left(\tilde{\boldsymbol{\nabla}}\tilde{\boldsymbol{\nabla}}\varphi : \tilde{\boldsymbol{\nabla}}\tilde{\boldsymbol{\nabla}}\varphi + \alpha_1 L^2 \tilde{\boldsymbol{\nabla}}\varphi \cdot \tilde{\boldsymbol{\nabla}}\varphi + \alpha_0 L^4 \varphi^2 \right) d\tilde{D} \tag{1.6}$$

Hence α_0 fixes the length scale over which variations are significant to move the *kernel function* from one to zero:

$$\alpha_0 L^4 = 1 \tag{1.7}$$

μL^2 fixes the relative weight on data (signal, σ^2) vs regularization (noise, ϵ^2):

$$\mu L^2 = 4\pi \frac{\sigma^2}{\epsilon^2} = 4\pi S/N \tag{1.8}$$

Finally α_1 fixes the influence of gradients:

$$\alpha_1 L^2 = 2\xi \tag{1.9}$$

where $\xi = 0$ if no penalization and $\xi = 1$ if penalization. ξ is a non-dimensional parameter close to one if the gradients are to be penalized with a similar weight than the second derivatives.

Weights on data

A weight μ_i can be assigned to each data d_i. This weight expresses the confidence you have in a particular data. It is expressed as a function of the signal-to-noise ratio and the correlation length:

$$\mu = \frac{\sigma^2}{\epsilon^2} \frac{4\pi}{L^2} \tag{1.10}$$

when $\xi = 1$. In Diva, the fourth column of the data input file, if present, allows to apply a different weight to each point.

Background field

Normally, interpolation (and extrapolation) works on anomalies with respect to a background field (Eq. 1.1). Diva allows you to work with different background fields, depending on the analysis you want to perform:

- no treatment is applied (the data you treat are already anomalies);
- the mean of data is subtracted from the data values;
- the linear regression (plane) is subtracted;
- additional subtracted semi-normed field ($\alpha_0 = 0$ and large L) obtained by two chained Diva executions [1].

In particular when no treatment is applied, with $\alpha_0 > 0$, the minimization forces the analysis towards zero when there are no data points in a distance comparable to L. This is coherent with the idea of an anomaly only.

[1] At now only available with the Graphical Interface version.

1.2.2 Error field

For any analysis method, it is important to have a quantification of associated errors. For example, you may want to use the relative error field to mask regions where you consider errors are too high.

We may expect the error field to be affected by:

1. the data coverage: the error field is expected to be higher where the data coverage is lower;
2. the noise on the data.

In Diva, error fields are calculated by analogy with Optimal Interpolation: since analysis in O.I. is equivalent to analysis with VIM and since error field of O.I. equals analysis of covariance fields, error field of VIM equals analysis (by VIM) of covariance fields.

In practice, the data input of the analysis tool is a vector containing the covariance of data points with the point in which the error estimate is to be calculated.

1.2.3 Kernel and correlation function

Kernel functions can be examined by analysing a single point with high signal-to-noise ratio and no background field.

The exact function in an infinite domain is given in blue. Differences between the two curves are only due to boundaries.

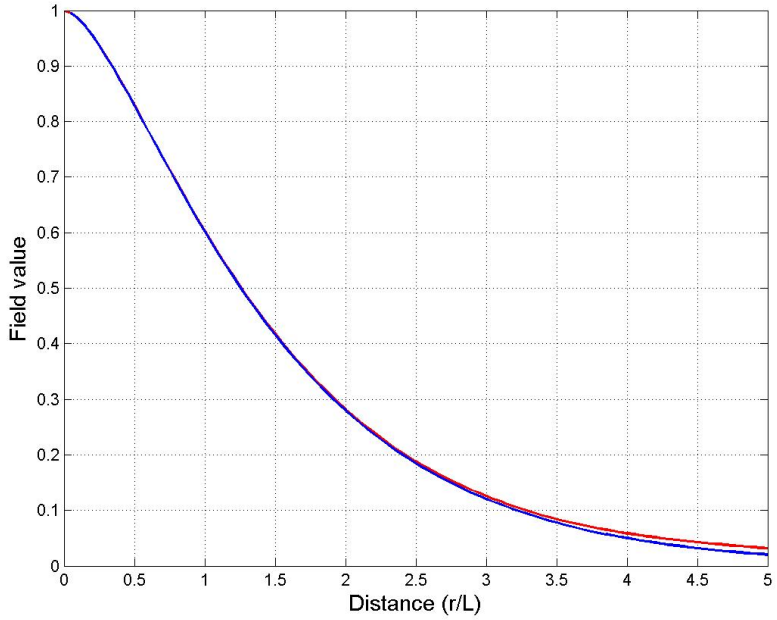


Figure 1.3: Kernel function (blue) and analysis of a single point (red).

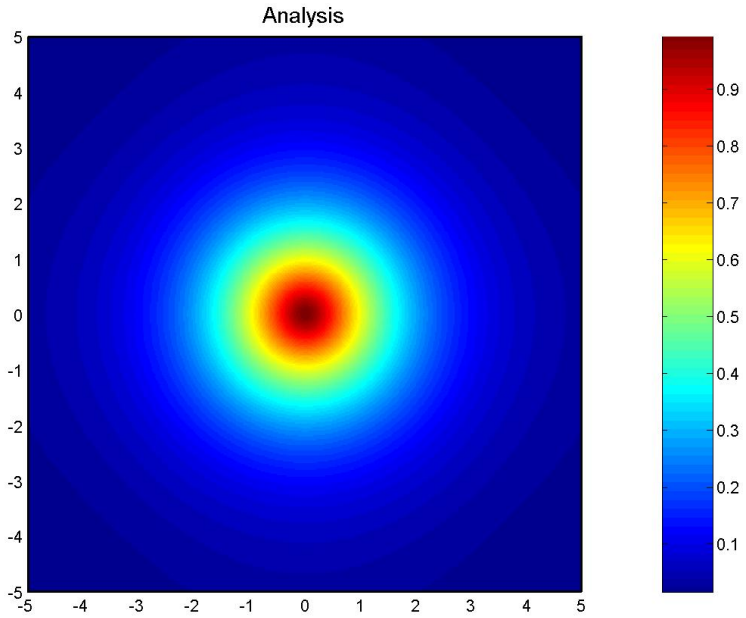


Figure 1.4: Result of a single point analysis with a unit anomaly and high signal-to-noise ration in a large domain.

The Kernel function can be used to calibrate Diva parameters $(\alpha_0, \alpha_1, \mu)$ so as to fit observed covariance functions.

1.2.4 Domain gridding

The minimization of 1.3 is actually performed by a Finite-Element Method, hence the need for generating a finite-element grid. Because the field is only defined in the water, the minimization also works only within the contours defining the coastline.

Thus the grid generation has to be consistent with the coasts existing in the considered region. The corresponding mathematical problem is referred to the *Constrained Triangulation*. More details about the finite-element grid will be given in section 3.1.3.

1.3 Comparison between the methods

Method	$\min(\varepsilon^2)$	3D	Multivar	Ops/image	$\varepsilon(\mathbf{r})$	*a priori*	C.V.	anisotropy
Cressman		$\star$	$\star$	$N_d N_a$		$w(r/L)$	(L)	$(\star)$
O.I.	$\star$	$\star$	$\star$	$N_d^3 + N_d N_a$	$\star$	$c(r/L)$	$L, \sigma^2/\mu^2$	$(\star)$
V.I.M.	$\star$	$(\star)$	$(\star)$	$N_a^{5/2}$	$\star$	$K(r/L)$	$L, \sigma^2/\mu^2$	$(\star)$
DINEOF	$(\star)$	$\star$	$\star$	$N_a^{5/4}$	$(\star)$	stat.	N	$\star$

Table 1.1: Characteristics of different methods of data analysis.

where

N_d : number of data points,

N_a : number of grid points for analysis,

N : number of EOFs,

L : correlation length,

σ^2/ε^2 : signal to noise ratio,

$\star$: available feature,

$(\star)$: available with some adaptations.

1.4 Additional tools

Diva software is provided with addition tools designed for parameters adjustment and validation of results. The tools from 1.4.1 to 1.4.3 are already available. The others are under development.

1.4.1 Generalized Cross Validation (GCV)

The Cross validation consists in setting data aside for a global error estimate. GCV applies analysis to random anomalies at the same locations as real data. It allows also estimating global errors and calibration of the correlation length L and the signal-to-noise ratio $\frac{\sigma^2}{\epsilon^2}$.

Please refer to Chap. 6 for more details.

1.4.2 Advection constraint

An advection constraint can be taken into account in the variational principle (1.4).

For this purpose, a velocity field has to be specified on a regular grid (similar to topography for 3D version). This will have the effect to increase correlation *along* streamlines and decreases *across* streamlines (fronts).

Be aware that error fields are in this case not equivalent to error fields of OI.

1.4.3 DINEOF

DINEOF stands for **D**ata **IN**terpolating **E**mpirical **O**rthogonal **F**unction.

It is designed to exploit repeated observations with missing data (*e.g.* a series of collocated satellite images with clouds). For more details about the DINEOF tool, the reader is invited to consult: [Alvera-Azcárate et al. (2005)],
[Alvera-Azcárate et al. (2006)],
[Beckers and Rixen (2003)],
[Beckers et al. (2006)] and
`http://ocgmod2.marine.usf.edu/DINEOF-welcome.html`.

1.4.4 Possible developments

Some aspects of Diva are still under development. Here are the improvements that will be developed in the future:

- 3D extensions: from surface to deep layers through EOF calculated from data set (DINEOF approach), horizontally VIM.

- Static stability constraints and other constraints.
- Multivariate approaches (easy in OI, but more difficult in VIM).
- Parameter calibration by cross validation in multivariate version.
- Data query modules, possibly through ODV.
- Strategy for combined analysis of satellite and *in situ* data.
- ...

2

Installation procedure

DIVA is a software designed to run under Windows as well as under Unix. The installation procedures are described for both operating systems.

2.1 Tcl/Tk

The graphical interface was created with the help of the language Tcl/Tk. Tcl stands for **T**ool **C**ommand **L**anguage and is an interpreted language which can work on Mac, PC and UNIX stations. Tk stands for **T**ool **K**it, a set of tools allowing the creation of graphical interfaces. For more details the following websites can be consulted:

- Tcl Developer Site:
  `http://www.tcl.tk/`
- Tcl/Tk Software download:
  `http://www.tcl.tk/software/tcltk/choose.html`
- Tcl/Tk 8.4 Manual: commands and libraries description:
  `http://www.tcl.tk/man/tcl8.4/`

2.1.1 Windows

There are two possibilities to generate the graphical interfaces under Windows using the language Tcl/Tk: `Cygwin` or `Msys`.

`Cygwin` can easily installed (or updated) from `http://www.cygwin.com/` by running `setup.exe` found there. `Cygwin` is a complete Linux-like environment, including the possibility to use Tcl/Tk.

Msys constitutes the minimal environment to be able to program in Tcl/Tk under Windows. The msys installation is described in the following.

Msys installation

1. Download the compressed file from
   http://prdownloads.sourceforge.net/tcl/msys_mingw8.zip
2. Unzip the msys_mingw8.zip (Example: in C:/).
3. Open the msys shell by double-clicking on the msys icon (MS-DOS command file).

Tcl/Tk installation

1. Download the most recent versions of Tcl and Tk from
   http://www.tcl.tk/software/tcltk/choose.html
2. Unzip tcl8xxx-src.zip and tk8xxx-src.zip within the directory[1]
   msys/home/username/
3. Copy the folders tcl8.x.xx and tk8.x.xx (contained in tcl8xxx-src and tk8xxx-src) into msys/home/username/
4. Open a msys shell and go into the directory
   msys/home/username/tcl8.x.xx/win
5. Type *configure* to create the makefile [2]
6. Type *make install* to perform the installation
7. Repeat the operations (5) and (6) in the directory
   msys/home/username/tk8.x.xx/win
8. Make sure the files tclshxx.exe, wishxx.exe, tclxx.dll and tkxx.dll are located in msys/local/bin. If it is not the case, copy them from the folder tcl8.x.xx

To check if your installation is completed, type *wishxx* in the msys shell (Example: type *wish84* if you are working with the version 8.4). You should get a command interpreter (Fig. 2.1).

Cygwin users: assuming that Cygwin is already installed on your computer, you can follow the previous steps by replacing msys/home/username/ by cygwin/home/username/.

[1] Where /username is your personal directory.
[2] This step and the following one may last several minutes, so please be patient.

2.1.2 Linux

Under Linux, the `msys` (or `cygwin`) installation is not required. Before installing a new version of Tcl/Tk, you are advised to test if your computer already dispose of a version compatible with the graphical interface. To this end, follow the instructions given in section 2.2.3.

If you get the fig. 2.3, then you do not need to install a new version of Tcl/Tk.
If the operations from section 2.2.3, following the procedure hereafter to install Tcl/Tk.

Installation of Tcl/Tk under Linux

1. Download the most recent versions of Tcl/Tk from
   `http://www.tcl.tk/software/tcltk/downloadnow84.html`
2. Uncompress and unpack the archives `tcl8.x.xx-src.tar.gz` and `tk8.x.xx-src.tar.gz` within the directory `/home/username/`:

   gunzip tcl8.x.xx-src.tar.gz
   tar -xvf tcl8.x.xx-src.tar
   gunzip tk8.x.xx-src.tar.gz
   tar -xvf tk8.x.xx-src.tar

3. Go into the directory
   `/home/username/tcl8.x.xx/unix`.
4. Type successively:

   configure to create the `makefile`,
   make to create a library archive `libtcl<version>.a` and an interpreter application called `tclsh` that allows you to type Tcl commands interactively or execute script files.
   make install to perform the installation of Tcl binaries and script files in standard places (you need to work as a root to have all the permissions).

5. Repeat the operation (4) in the directory
   `/home/username/tk8.x.xx/unix`.
6. Add `tclsh` and `wish` to the path: edit bashrc and add
   `:/home/username/tcl8.x.xx/unix/:/home/username/tk8.x.xx/unix`

2.2 Diva software installation

The two versions of Diva (Command Line [**CL**] and Graphical User Interface [**GUI**]) are installed by downloading `diva-4.1.zip` from `http://modb.oce.ulg.ac.be/modb/diva.html` and unzipping where you want to install Diva on your hard disk (Example: in `C:`).

The directory tree you will obtain is showed on Fig. 2.2.

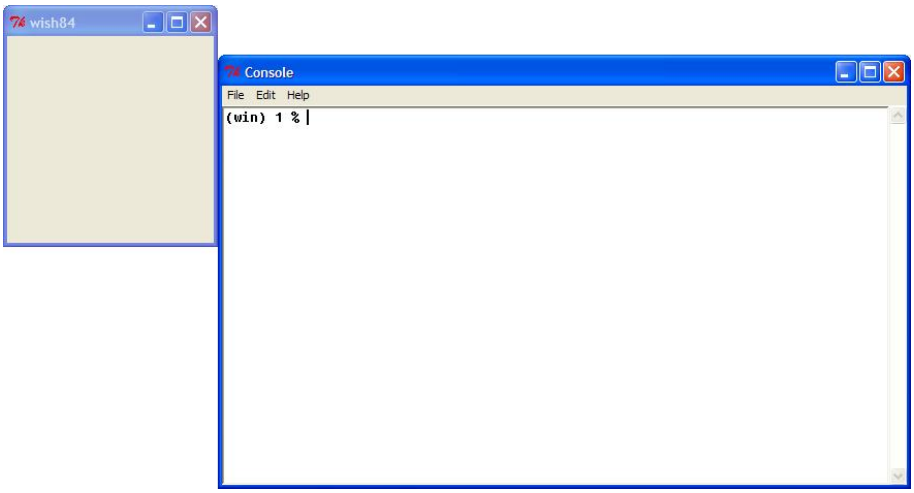


Figure 2.1: Wish interpreter console.

2.2.1 Diva-4.1 content

The folder `diva-4.1` contains five main subfolders:

- `./bin` is where the executables are located,
- `./divastripped` is the working directory for the CL version (see section 3).
- `./doc` contains the publications and the documentation. You can also find the corresponding BibTeX entries in the file `DivaPublications.bib`.
- `./GUIwork` is the working directory for the GUI version (see section 3).
- `./src` contains the source code for the Diva analysis (folder `Fortran`) and the graphical interface (folder `Tcl`). It also contains Matlab scripts for plotting the meshes and the results of the analysis (folder `Matlab`).

2.2.2 Command Line version

The Command Line version is directly usable through a shell in the `divastripped` directory.

2.2.3 Graphical User Interface version

To finish your Diva GUI installation, you only need to set the paths according the location of the `diva-4.1` folder on your hard disk:

1. Go to `diva-4.1/src/Tcl`, edit the file `main.tcl` and set the path to your `diva-4.1` directory. For example, if you installed `diva-4.1` in `C`:
   `set env(DIVA_GENERAL_DIR) c:/diva-4.1`

```
|-bin                (*.a : compiled executables)
|---SGI              (precompiled version for SGI)
|---Unix             (precompiled version for Unix)
|---Win32            (precompiled version for Windows)
|-divastripped       (command line executions and diva* scripts)
|---divawork         (working directory)
|---input            (input files)
|---meshgenwork      (mesh generation working directory)
|---output           (output files)
|-----ghertonetcdf   (whre the NetCDF files are stored)
|-----meshvisu
|-----plot           (where the plots are stored)
|-doc                (documentation and related publications)
|-src                (sources)
|---Fortran
|-----Calc           (diva core)
|-----Extensions     (for 3D in GUI)
|-----Mesh           (mesh generation)
|-----NC             (NetCDF)
|-----NoPlplot
|-----Plplot         (visualization routine)
|-----Util           (Utilities)
|---Matlab           (some post-treatment in \matlab)
|-----CL             (Command Line version)
|-----GUI            (Graphical User Interface version)
|---Tcl              (files to generate the GUI)
|-GUIwork            (working place for GUI)
|---ANALYSIS         (analysis files)
|---BASE             (data base)
|---BAT              (bathymetry files)
|---CONFIG           (configuration files)
|---CONT             (contour files)
|---DATA             (data files)
|---DEPTH            (depth files)
|---DES              (data base description files)
|---MESH             (mesh files)
|---TMP              (temporary files)
|-install            (elements for the software installation)
```

Figure 2.2: Diva-4.1 directory tree.

2. In `diva-4.1/GUIwork/DES`, edit the file `liste.DES` according to the location of `diva-4.1` in your hard disk.

   Example:

   c:/diva-4.1/GUIwork/BASE/demo.BOT.pdf
   c:/diva-4.1/GUIwork/BASE/demo.BT.pdf
   c:/diva-4.1/GUIwork/BASE/demo.CTD.pdf

3. (a) For **Windows**:

   Open an (`msys`) shell and go to the directory `diva-4.1/src/Tcl`.
   Type *wishxx main.tcl* (where xx refers to the version you are using) to activate the interface.

   (b) For **Unix**: in the directory `diva-4.1/src/Tcl`, type *wish main.tcl* to activate the interface.

4. If the installation succeeds, you will get a windows like Fig. 2.3.

Figure 2.3: Graphical interface menu bar.

2.3 Compiling

The executables produced by the Fortran programs should be located in `diva-4.1/bin`. In this purpose, you are offered two possibilities:

1. Use the precompiled versions of the executables for Windows, Linux and SGI which are provided in `diva-4.1/bin/Win32`, `diva-4.1/bin/Linux` and `diva-4.1/bin/-SGI`, respectively. Simply copy the executables corresponding to your operating system to `diva-4.1/bin`.

2. Compile the Fortran programs, for example if you need to modify internal parameters to increase memory allocation.

   Execute the file `compileall` located in `diva-4.1/src/Fortran`. This file makes the compilation of the Fortran scripts in each folder: `Mesh`, `Util`, `NC`, `(No)PlPlot` and `Calc`. The `makeall` should not be modified.

   You only have to adapt the files `makediva` (in `Calc`) and `makeall` (in the other directories) according to the Fortran compiler and the compilation options you are using. Examples of `makediva` and `makeall` files are provided with the suffixes `SGI`, `LINUX` and `WIN`.

   All the executables (`.a` files) are automatically moved to the directory `diva-4.1/bin` after the compilation.

2.3.1 Tcl/Tk programs

As Tcl/Tk is a interpreted language, you do not have to compile it. The modifications you may have made will be taken into account once you will have launched the Diva GUI.

2.4 Creation of shortcuts

Here are the procedures to create shortcuts for GUI and for CL on your desktop.

2.4.1 Windows

- Copy `diva.ico`, `divacl.ico`, `diva.bat` and `.profile` (all found in `diva-4.1/-install`) into the `msys` directory.
- Edit `diva.bat` and change the line starting by `start sh -login -c "cd` to adapt the path to the directory in which you installed the Diva `.tcl` files.
  Example:

  :startsh
  start sh –login -c "cd c:/diva-4.1/src/Tcl; wish84 main.tcl & "
  rem exit

  and save the modified `diva.bat` file.
- Edit `.profile` and modify the third line according to your configuration.
  Example: cd *c:/diva-4.1*
- With explorer, create shortcuts to `msys/diva.bat` and `msys/divastripped.bat`, and move them to the desktop.

- Right-click on the desktop-shortcut-Properties:
  in General: use `diva-4.1 GUI` instead of shortcut ... as text for the GUI version,
  `diva-4.1 CL` for the CL version.
  in Shortcut: change icon, browse to msys and chose `diva.ico`,
  `divacl.ico`.

2.4.2 Linux

- Right-click on your desktop and select *Create launcher*,
- Type = application,
  name = DIVA (as you wish),
  command = path to the `wish` executable + path to `main.tcl`.
- In permission: allow executing file as program.

2.5 PlPlot library

The PlPlot library is a set of Fortran routines that allow you to obtain graphical outputs without resort to external softwares, such as Matlab or NcView. We describe hereinafter the procedure for the installation.

1. download the most recent version of PlPlot from
   `http://plplot.sourceforge.net/`
2. uncompress and unpack the archive `plplot-5.7.1.tar.gz`

   gunzip plplot-5.7.1.tar.gz
   tar xvf plplot-5.7.1.tar.gz
3. in the directory `plplot-5.7.1`, type:

   ./configure –prefix=MYPREFIX, where MYPREFIX stands for the installation prefix (Example: `/usr/local/plplot`)
   make
   make install
4. in the directory MYPREFIX/share/plplotVERSION/examples, type:

   make
   ./plplot-test.sh to see some examples.

   The complete installation procedure is described in the file `INSTALL`.

3 Diva Use

3.1 General

We describe here what are the ingredients needed by Diva to perform an analysis and the way the input files have to be created.

3.1.1 Data

The data file contains three (or four) columns: `X | Y | value | (relative weight)`. If the number of column is three, the fourth column is assumed to take the value 1.

```
----
25 25 5
25 75 10
75 75 5
75 25 10
----
```

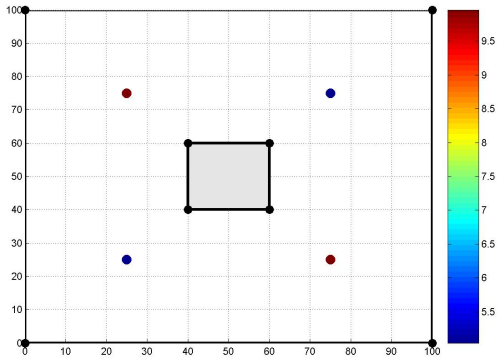


Figure 3.1: Example of a data file and its graphical representation.

3.1.2 Contour

The contour files are defined this way:

1. The first line indicates the number of contour in the region of interest (say M).
2. The second line tells the number of points in the first contour (say N_1).
3. The next N_1 lines are the coordinates of the points of the first contour. The convention for the contour is that *the land is on the right when you follow the points successively*. The contour is automatically closed, meaning that the last point of a given contour has not to be the same as the first one.
4. The following line is the number of points of the second contour (say N_2).
5. ...
6. The last N_M lines are the coordinates of the points of the last contour.

```
----
2
4
0 0
100 0
100 100
0 100
4
40 40
40 60
60 60
60 40
----
```

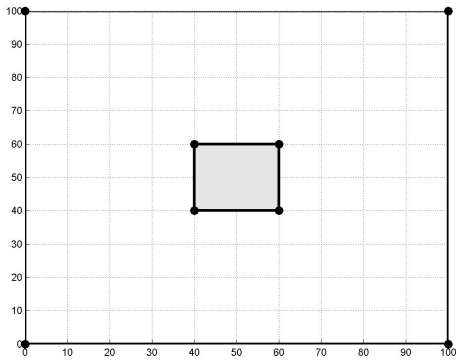


Figure 3.2: Example of a contour file and its graphical representation.

3.1.3 Finite Element mesh

As explained in the first chapter, the solution of (1.3-1.4) is computed with the help of a finite-element method. The mesh is built using the coastline provided by the user.

The main parameter ruling the shape of the mesh is the size of the element L_e. This length scale should be small compared to the scales of interest.

As in reality each element is the combination of three smaller, third-order FEM, the actual resolution is rather $L_e/3$.

When deciding the characteristic length of the elements, you should be aware of the following facts:

- Contour segments cannot be much smaller than finite element length ($L_c > L_e$). If your contour is too fine, the tool `divacck` can be used in order to reduce the contour resolution.
- The correlation length should be larger than L_e (otherwise the grid is too coarse compared to the signal to resolve).
- When requesting a too fine mesh, you might need to recompile diva to increase memory allocation (programs written in Fortran 77).

3.1.4 Parameters

In each step of a Diva analysis, many parameters can be changed according to the particular case you are treating. If you work with the CL version, these parameters have to be entered in the file `param.par` (see file 3.1.1). If you work with the GUI version, the parameters are chosen as one goes along the analysis.

Here is a summary of the parameters you may have to modify.

General

Coordinates system: kilometers or longitude/latitude.

Mesh

Uniform or non-uniform mesh: density mesh constant or varying within the domain. If non-uniform mesh, a density file (`coast.cont.dens`) has to be provided.
Reference length: length of the side of the triangular elements.
Surface coefficient: shape of the triangular element.
Smooth number: smoothing of the mesh elements.

Analysis

Correlation Length: radius of influence of a data at a given point.
Signal-to-noise ratio: relative importance of the signal variance with respect to the noise variance.
Reference field: background field which is subtracted from the data field.
Variance of the background field.

Output grid parameters

`xori yori dx dy nx ny`.
Exclusion value: value used to fill the ouput matrix when a point corresponds with land.

```
# Lc: correlation length (in units coherent with your data)
10
# icoordchange (=0 if no change of coordinates is to be performed;
# =1 if positions are in degrees and if you want to use real distances)
1
# ispec (output files required)
3
# ireg (mode selected for background field: 0=null guess;
1=mean of data; 2=regression plan if at least
3 non-aligned data provided)
1
# xori: x-coordinate of the first grid point of the output
15
# yori: y-coordinate of the first grid point of the output
66
# dx: step of output grid
0.2
# dy: step of output grid
0.05
# nx: number of grid points in the x-direction
300
# ny: number of grid points in the y-direction
300
# valex (exclusion value)
-999.0
# snr signal to noise ratio of the whole dataset
30
# varbak: variance of the background field. If zero, no error fields are produced.
# If one, relative errors are obtained
0
```

Example file 3.1.1: *param.par*

The example file `param.par` can be used as a model for your own use. The signication of the parameters is discribed in the file itself and are also made explicit in the present section.

3.2 Command Line version

The CL version allows for batch processing and customization, but is not practical for beginners. This version is much better suited for repeated and automated treatment and will also benefit from latest developments earlier.

Important remark: when working under Unix, you may have to convert the compilation scripts (`makediva` and `makeall`) and the *procedures* files (files with a name starting by `diva*` located in `diva-4.1/divastripped`).

To perform the conversion, use the command *dos2unix* [1] followed by the name of the file you want to convert.
Example: *dos2unix divacalc*.

3.2.1 Working directories

The Command Line version is contained in `diva-4.1/divastripped`. Four subdirectories exist inside `divastripped`:

1. `input`: data, contour(s) and parameters needed by Diva to perform an analysis.
2. `meshgenwork`: mesh generation working directory.
3. `divawork`: the working directory of the CL version, where the Diva calculation is performed.
4. `ouput`: results for the user.

3.2.2 Input files

The data file (`data.dat`), the coastline file [2] (`coast.cont`) and the file that indicates the points where you want an analysis (`valatxy.coord`).

Several examples are provided in the Diva distribution.

[1] This command should be available within you `msys` or `cygwin` distribution.
[2] The generation of coastline file is explained in section 3.5.1.

3.2.3 Output files

A complete run of Diva generates numerous output file. If you are only interested by the analysis results, here are the files you shall have a look at:

`fieldgher.anl` and `errorfieldgher.anl` are respectively the analysis and the error fields (in gher format) on the regular grid specified by `x/yorig`, `dx/y` and x/yend. `fieldascii.anl` and `errorfieldascii.anl` are the same as `fieldgher.anl` and `errorfieldgher.anl`, but in ascii format.

`valatxyascii.anl` and `erroratxyascii.anl` give respectively the values of the analysis and the error fields at the points specified in the file `valatxy.coord`.

`fieldatdatapoint.anl` and `erroratdatapoint.anl` are respectively the analysis and error fields computed at the data points (*i.e.* the points from `data.dat`).

The other files generated by Diva are described in the additional tools in the following section.

3.2.4 Running an analysis

Simple run

Copy the data you want to process into the subfolder `divastripped/input` with the following names: `coast.cont`; `data.dat`; `valatxy.coord`; `param.par`.

Set your command line into the folder `Divastripped` and execute by typing: *divadress*. This will execute the four following procedures:

`divaclean`: cleans up the working directories by removing the intermediary `fort.*` files from `divawork` and `meshgenwork`, and the output files from `output`.
+ `divamesh`: creates the finite element grid.
+ `divacalc`: based on the existing mesh, performs the analysis.
+ `divaqcbis`: simple outlier detection using generalized cross-validation output from `divacalc`.

Additional tools

A few utilities are provided to help you calibrate the signal/noise ratio and/or the correlation length.

- `divafit`: a script that uses `./input/data.dat` for a direct fitting of the covariance function.

  In output file `./output/paramfit.dat`, the best estimates are given and could be used as parameter values for running Diva. Estimates of the correlation length are rather robust while those of the signal-to-noise ratio are neither precise nor robust, especially for large values.

The fit needs a sufficiently large data set.

Output file `covariance.dat` is the data-based covariance function (column 1: distance between points, columns 2: covariance, column 3: number of data couples used to estimate the covariance).

Output file `covariancefit.dat` allows looking at the fitted covariance function (column 1: distance between points, column 2: data-covariance, column 3: fitted covariance)

- `divagcv`: a script that exploits a new Diva module (`gcvfac`), analysing random fields to assess the generalized cross validator (GCV).

  Input to the module is the number of random estimates required (the larger the value, the more robust the estimator. Default value is 5, unless you change in `divadress`.)

  The script `divagcv` is an example how to minimize the estimator by changing the S/N value, but could be adapted to optimize other parameters as well.

  Input file `./input/gvcsampling.dat` contains the list of values for S/N on which to try the estimator (typically around the values provided by `divafit`.

  During the `divagcv` execution, error-field calculations are disabled to reduce computing time.

  Output file `./output/gcv.dat` contains the GCV estimator GCV (column 1: S/N, column 2: GCV, column 3: data anomaly variance) and `./output/gcvsnvar.dat` the best new estimate for the S/N and `VARBAK` parameters.

- `divacck`: contour check. In output `./output/coast.cont.checked`, you find a thinned contour based on the length scale.

- `divaqcbis`: simple detection of outliers, listed in `./output/outliersbis.dat`, with a last line giving an indication if this is normal or not. Be aware that this detection is only valid as long as your S/N and `VARBAK` parameters are correct. Systematic unusual numbers of outliers might indicate a problem in the parameter calibration (S/N, L) and/or error field calculation (`VARBAK`).

- `divaqc`: advanced outlier detection (see Section 6.3 for a detailed description).

Finally the typical execution chain for a complete Diva run can be summarised as:

1. Preparation of a first `./input/param.par` (including coordinate change and reference field definition.)
2. `divafit` for a first guess of the parameters;
   adaptation of `./input/param.par` according to the output file `paramfit.dat` and creation of `./input/gcvsampling.dat`
3. `divacck` for testing the contour.
4. `divamesh` to generate the mesh.
5. `divagcv` for improved estimates of parameters;
   adaptation of `./input/param.par` according to the values given in `gcvsnvar.dat`
6. `divadress`: chains `divaclean, divamesh, divacalc, divaqcbis` for analysis as well as error field whenever `VARBAK` is not zero;
   detection of outliers with the help of `divaqcbis`.
7. `divaqc` additional quality check of data, listed in `./output/outliers.dat`

3.3 Graphical Users Interface

The Diva GUI is made of four main menus: `File`, `View`, `Options` and `Help` (fig. 2.3). `File` and `View` are both made of three sub-menus: `Data`, used for the data extraction from a data base, `Mesh`, which generates the finite elements mesh, and `Analysis`, which performs the field reconstruction using the variational inverse method (VIM). The menu `Options` permits the choice of the coordinates system (Coordinates), the directories and the language (at now, English and French are available).

3.3.1 Working directory

The `GUIwork` directory is made of the following folders: ANALYSIS (analysed and error fields), BASE (data base files), BAT (bathymetry), CONFIG (pre-defined configurations), CONT (contour files), DATA, DEPTH, DES (data base description files), MESH and TMP (temporary files).

To easily identify the different kinds of files and to be in agreement with the interface conventions, it is suggested to use the following extensions for the file names:

`.anl` for the analysis files,

`.bat` for the bathymetry files,

`.conf` for the configurations files,

`.cont` for the contour files,

`.dat` for the data files,

`.des` data base description files and

`.mesh` for the mesh files.

3.3.2 The `Data` menu

Input:

- the *Data Base description file*: it indicates which data file names are used to search for the extraction of data.
- *Period (mmdd)*: the days of the year when you want to extract data. For example from 0901 to 1231 means from the 1st of September to the 31st of December.
- *Depths*: the depths where the data are extracted.
- *Contour description file*: describes the position of the coastline.
- *Position*: specify the longitudes and latitudes to select a rectangular domain.
- *Quality criterion*: two types of flags are introduced: one for the whole profile (profile quality flag) and one for each particular observation (observation quality flag). For more details, please consult the Diva Help menu.

Output:

- the *Generic data output file name*: the generic name of the extracted data files.

Commands:

- *Load* and *Save configuration*: once you have entered all the file names and the parameter value, it is useful to save your configuration, in order to gain time.
- *OK*: performs the data extraction.
- *Display Data File*: allows the user to check the output file.

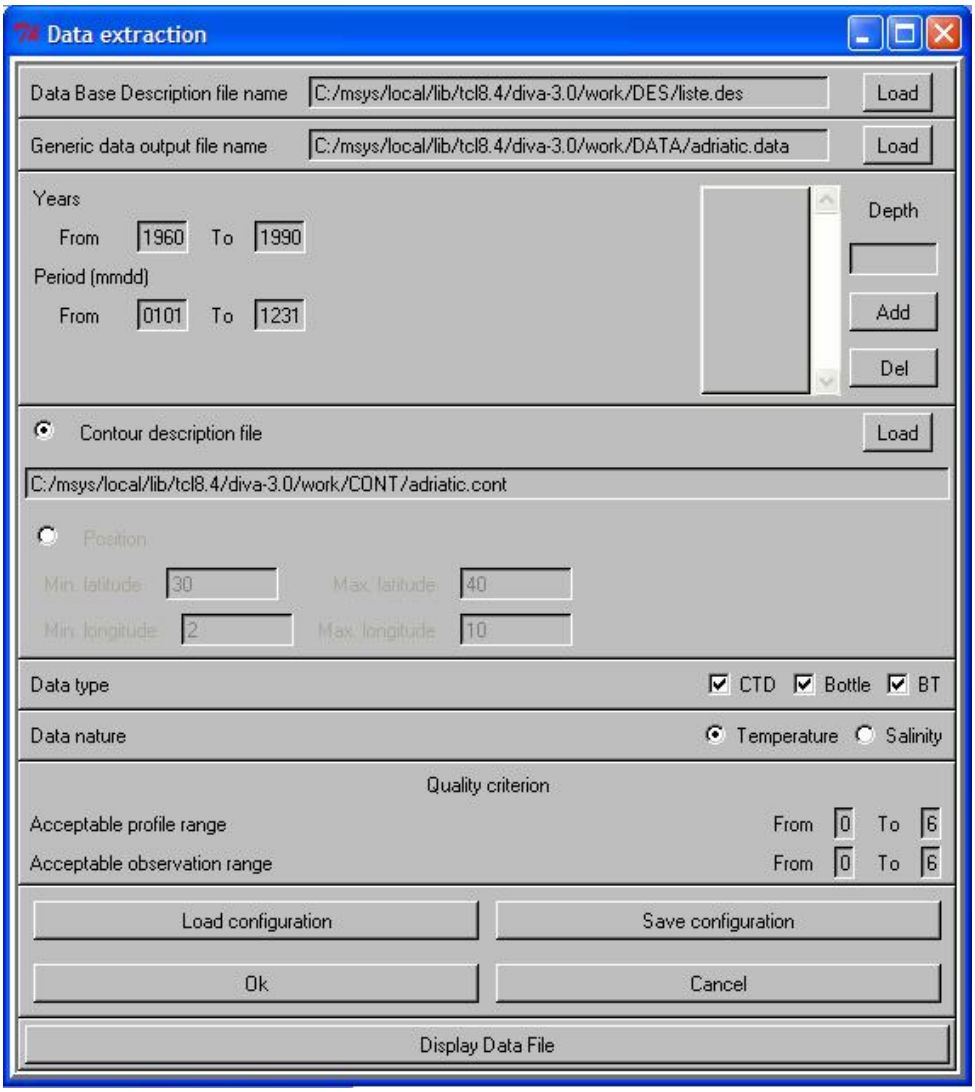


Figure 3.3: Data extraction window.

3.3.3 The `mesh` menu

Input:

- two- or three-dimension analysis,
- *Contour description file*,
- *Bathymetry file name* (3D case only): file that contains the bathymetry of the region where you want to work (regular grid in the GHER format).
- *Depth file name* (3D case only): contains a list of depths and the corresponding data files.
- *Uniform* or *Non-Uniform mesh*: in the "Non-Uniform mesh" case, a different density mesh can be used in some parts of the domain.
- *Reference length*: fixes the length of the side of the triangular elements touching an outline.
  Surface coefficient: controls the shape of the triangular element; the value 1.5 provides triangles close to equilateral triangles.
  Smoothing number: leads to a more regular shape of the elements; the value 3 is suggested.

Output: The *Mesh file name* specifies the generic name of the file that will be generated through the mesh process. The software creates three files:

1. `.../name.mesh` contains the topology of the mesh,
2. `.../name.mh4` contains the number of vertex and interface nodes and the number of meshes,
3. `.../name.mh5` that takes into account the depth in the 3D case.

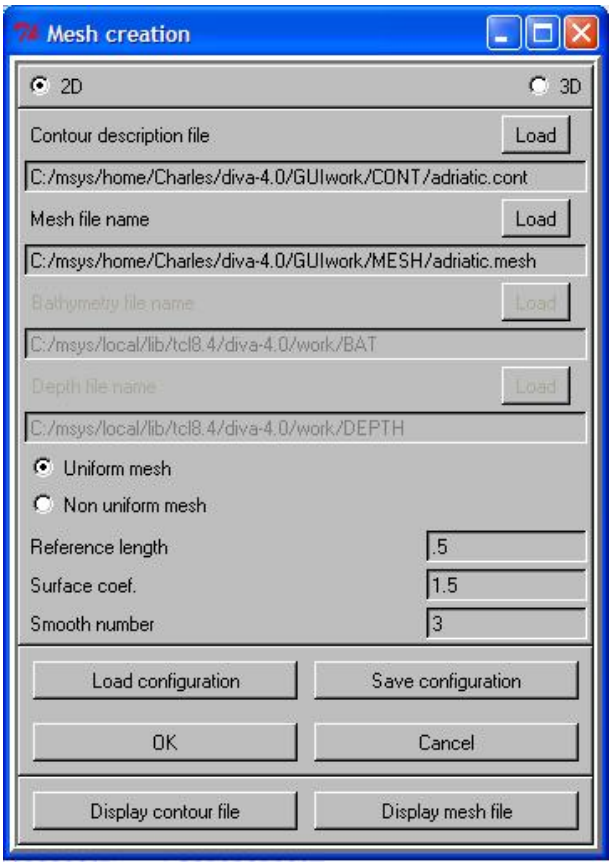


Figure 3.4: Mesh creation window.

3.3.4 The `Analysis` menu

Input:

- Two- or three-dimension case,
- *Data*, *Depth* (3D only), *Mesh* and *Bathymetry* (3D only) *file names*,
- *Compute error*: computation of the statistical error map of a field.
- Reference field: treatment to be applied on the data located in the mesh: (a) *None*, (b) subtraction of the *Mean Value*, (c) subtraction of the *Linear regression*, (d) *semi-normed analysis*.
- *Characteristic length* (L): interpreted as the radius of influence of a data at a given point;
  Signal/Noise ratio (σ): (variance of signal)/(variance of noise) ratio.
- α_0, α_1 and μ: directly set or computed from L and σ according to

$$\alpha_0 = \frac{1}{L^4}, \quad \alpha_1 = \frac{2}{L^2} \quad \mu = \frac{4\pi\,\sigma}{L^2}.$$

- Output grid parameters: *phi/lambda origini/end*, *dphi*, *dlambda*: limits of the grid where the resulting reconstruction is written.
  The *Exclusion value* is used to fill the output matrix when a cell (a point) corresponds with a point on land.
  Selected depth: the data for a given depth located in the list of data file name are reconstructed.
- *Transient results*: determines the amount of information appearing to the screen while computing the data reconstruction. Goes from 1 (minimal information) to 5 (maximal information).

Output:

1. The analysed field is stored in the `.anl` file.
2. The error field is stored in the output file suffixed by `.error`.

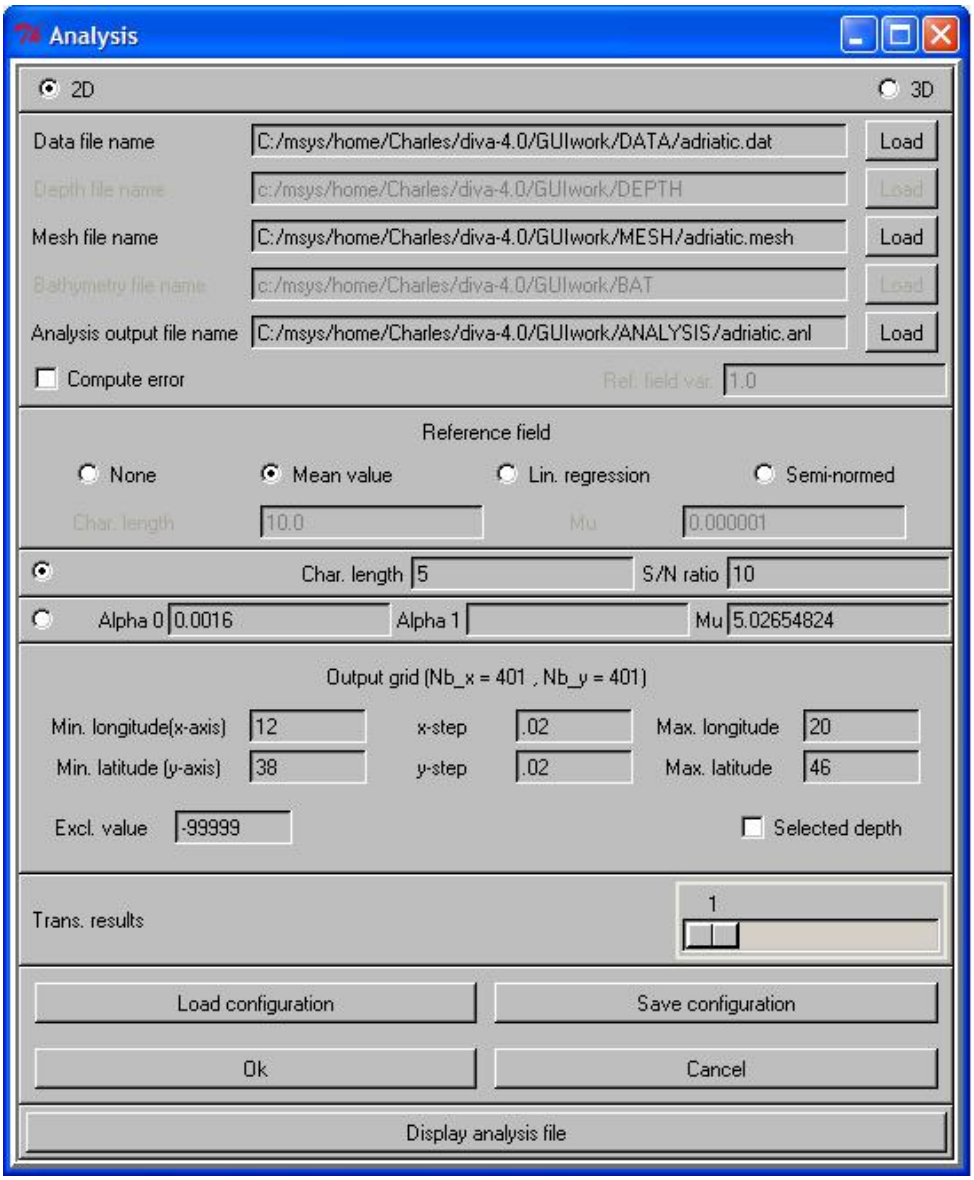


Figure 3.5: Diva analysis window.

3.4 The Options menu

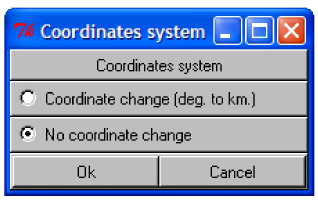


Figure 3.6: Coordinates system options.

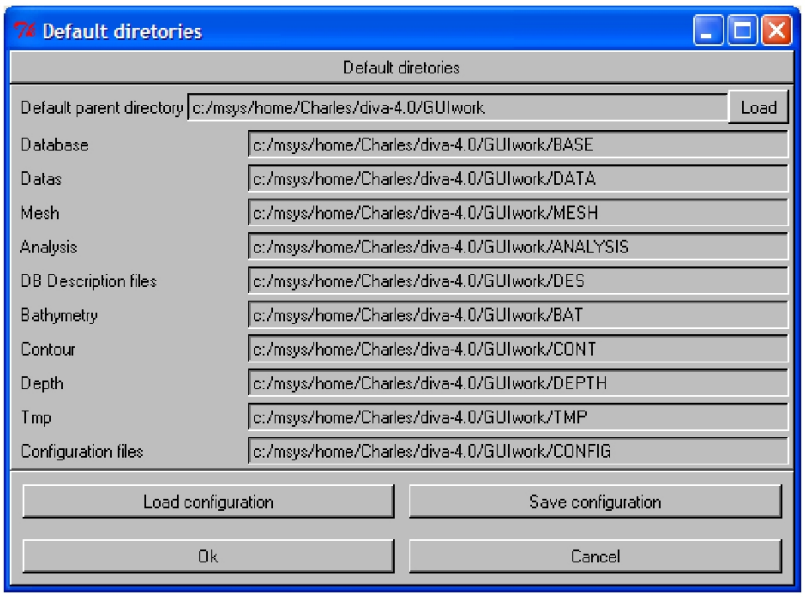


Figure 3.7: Default directories choice.

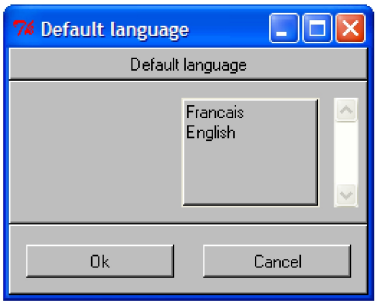


Figure 3.8: Language preference.

3.5 Generation of contour and topography files

This section is dedicated to the different methods available to create the contour (`coast.cont`) and topography (`coast.cont`) files necessary to perform an analysis with Diva.

3.5.1 Coastline generation

As seen in the first chapter, a relevant asset of Diva is the fact that it takes into account the real coastline and topography of the region of interest. We explain herinafter the ways

you can produce a correct coastline file.

The description of the coastline structure is provided in section 3.1.2.

By hand

The first possibility is to build your file *by hand*: having at your disposal the location of different points of the coast (longitude, latitude), you create a file containing M contours, with the $i-th$ contour having being made up of N_i points ($i = 1, 2, \ldots, N$).

Be aware that some cases (fig. 3.9) are to avoid: problems arise when crossing occur in the contour. Also note that, as the contours are automatically closed by Diva, the segment joining the first and the last point may generate errors.

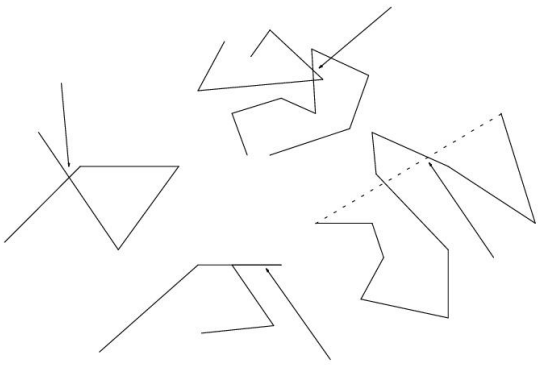


Figure 3.9: Example of improper contours. Left: crossing of two segments of a same contour; up: crossing of two different contours; right: first and last points of the contour generate a segment that crosses the other parts; down: two contours have a common segment.

Remember that you can use the tool `divacck` for checking and thinning of contours.

From topography

`divacont` uses `topo.grd` and `TopoInfo.dat` (see Section 3.5.2 to see how to generate them) from directory `./input` to make contour files `coast.cont` in directory `./output`

`divacont` creates a contour for each level defined in `contour.depth` (one value per line, in directory `./input`) A series of `coast.cont.100nn` will thus be created (`nn=01` corresponing to the first line), each of those can later be copied into a `coast.cont` for an analysis at the level of your choice.

In addition `coast.cont` is created with a default value $z = 0$.

From a mask

Simply look at `contourgen.f` and create the mask as you wish. Alternatively you can create a pseudo-topography with adequate pseudo-depth at which you draw the contour.

Mesh refinement

For all coastlines, a *by hand* approach allows to refine regions with increased finite element resolutions. To use this, create a `coast.cont.dens` file that contains

```
NR                    : number of subregions
L1 N1                 : length scale and number of points defining the subregions
x1 y1                 : coordinates
.. ..
xN1 yN1
L2 N2                 : length scale and number of points defining subregion 2
...
```

Subregions do not need to define real contours, only parts of the space are defined in which resolution is different.

Place this file into `./input` and perform as usual. Be sure you eliminate the file for other applications.

3.5.2 Topography generation

By hand

Create a gridded file (in the same format as the analysis fields `fieldgher.anl`) of topography and call it `topo.grd`. The information on the grid's geographical dimensions are to be placed in `TopoInfo.dat`.

The grid is simply an array $(i = 1, \ldots, M)$ and $(j = 1, \ldots, N)$, where the coordinates of the grid nodes are $x_i = x_1 + (i-1) * dx$, $y_j = y_1 + (j-1) * dy$. The file `TopoInfo.dat` contains simply

```
x1
y1
dx
dy
M
N
```

Look into `dvdv2diva.f` in `./src/Fortran/Util` how to write such files from within a Fortran code.

Topography extractor

From `https://idbms.navo.navy.mil/dbdbv/dbvquery.html`, select `Area` and desired spacing, then submit. Chose `CHRTR ascii format` and save resulting file as `topo.asc` into directory `./input`.

Execution of `dbdb2diva` in `divastripped` directory will then create `topo.grd` and `TopoInfo.dat` in `./output`. You can use matlab file `topovisu.m` to visualize the topography.

Then for automated contour generation (see subsection 3.5.1), move these files into `./input`

Topography generation with Diva

You can use Diva itself to interpolate a topography from isolated depth measurements. Such point data can be extracted for example from

`http://topex.ucsd.edu/cgi-bin/get_data.cgi`

With such a series of data `(x y depth)` saved in `topo.dat` an execution of `divatopo` in directory `./divastripped` creates `topo.grd` and `TopoInfo.dat` in `./output`. The analysis and output grid is based on the `param.par` found in the `./input` directory.

All topographies, be they created *by hand*, read or analysed, can then be used for contour generation (section 3.5.1) by moving the gridded topography `topo.grd` and its metainformation `TopoInfo.dat` into directory `./input`

3.6 Display of outputs

3.6.1 Matlab

Tools to display the contour, data, mesh, analysis and error fields are provided in the directory `diva-4.1/src/Matlab`. They are separated in two subdirectories: `CL`, for the Matlab programs adapted to the command line version, and `GUI`, for the graphical interface.

The operations performed by the Matlab programs are the same in both directories, but the input file names may be different. The plots will be exported either in *jpeg*, or in *encapsuled postscript (eps)* formats (choose between parameter `format_out` = 1 for *jpeg* format, `format_out` = 2 for *eps* format).

Four scripts allow you to create your plots:

1. `diva_contour.m`,
2. `diva_mesh.m`,
3. `diva_analysis.m`,
4. `diva_error.m`.

These four operations can be summed up by running `diva_plot_all.m`. Remember that these Matlab scripts aim to show you how to create plots from files generated by Diva. Therefore do not hesitate to modify them and adapt them to your own need.

With `diva_analysis_mask.m`, you have the possibility to plot the analysis field with a mask corresponding to the points where the error field has a value lower than a chosen

value. You just have to specified the higher bound allowed for the error (Example: `treshold` = .25) results that the analysis field will be plotted only at the location where the relative error is lower than 25 %.

Subroutines: `uread.m` performs the reading of files written in GHER format (binary). These files are opened and closes with the help of `gzopen.m` and `gzfclose.m`, respectively. These three files do not need to be modified.

Plots for CL version

They are designed to create plots by reading information in the subdirectories of `GUIwork`. By default, the plots are saved in `GUIwork/PLOT`.

To make a plot, you just have to indicate the name of the case you treating (Example: *adriatic*) and run the corresponding Matlab file. The program will load information required to perform the plot in the other directories.

Example: `adriatic.dat` in the `DATA` directory, `adriatic.mesh`, `adriatic.mesh.mh2` and `adriatic.mesh.mh4` in the `MESH` directory, etc.

If you are working with another convention for the file names, you just have to edit the `.m` files according to the names you have chosen.

The names of the figures are automatically determinated (Example: `adriatic_mesh`), but you can modify them easily by changin the line(s) beginning by
`fileout1=[Name,'_...'];`.

Plots for CL version

As the name of the input files are always the same (`data.dat` for the data file, `coast.cont` for the contour file and `param.par` for the parameter), you do not need to adapt the names of the input files. By default the suffix of the figures created is `output` (Example: `output_contour`),

3.6.2 NcBrowse and Ncview

Diva provides analysed and error fields under the NetCDF (network Common Data Form) format. It is an machine-independent format to represent scientific data. For more details, please consult `http://www.unidata.ucar.edu/software/netcdf/`.

The files written in this format can be easily displayed with the following tools:

- *NcBrowse* (Linux and Windows) available at
  `http://www.epic.noaa.gov/java/ncBrowse/`
- *Ncview* (Linux and Windows + Cygwin) available at
  `http://meteora.ucsd.edu/~pierce/ncview_home_page.html`

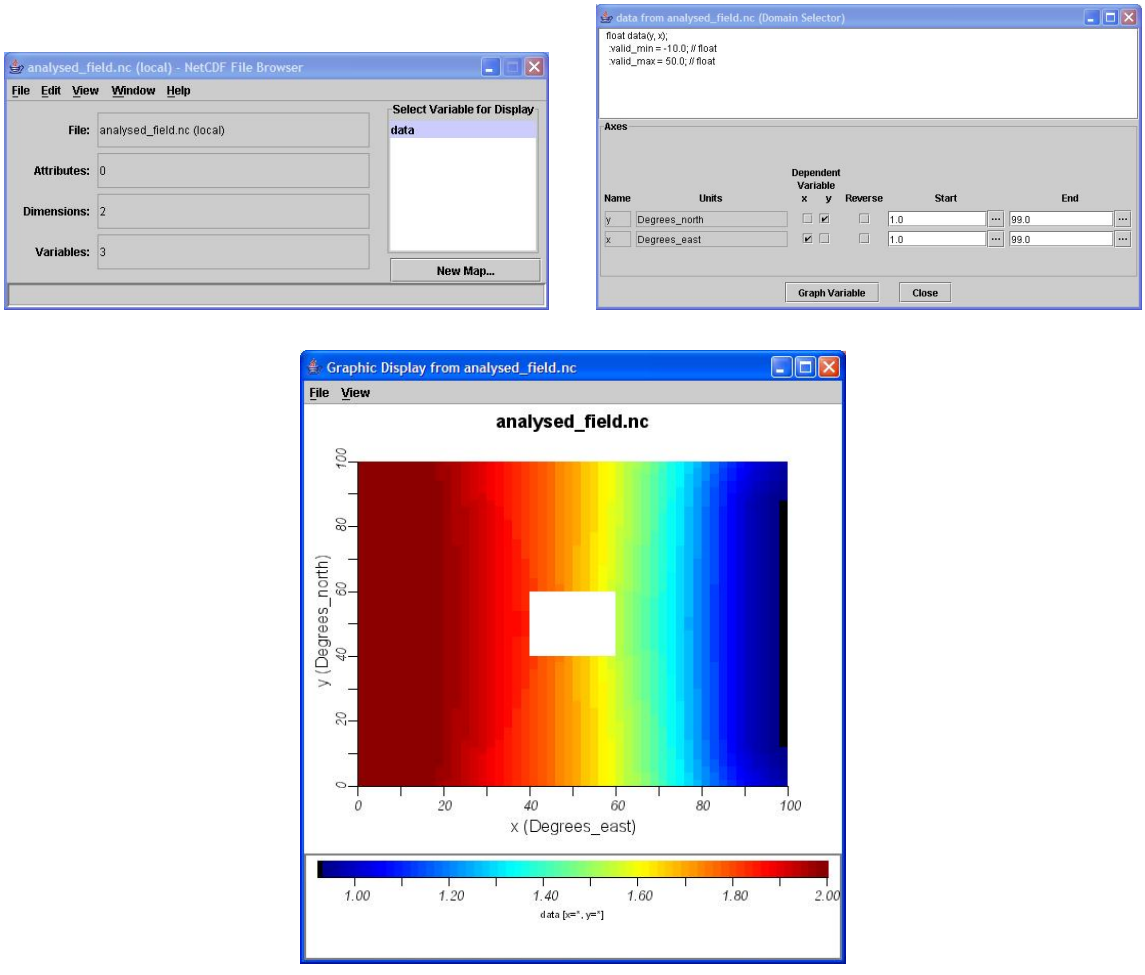


Figure 3.10: Plots of results with NcBrowse.

4

Test cases

THE test cases are idealized situations designed to show some properties of the Diva analysis and to allow the use to observe the influence of the variation of parameters.

4.1 Island

This case shows the influence of the data density and of the influence of a specific contour (an island) on the analysis.

4.1.1 Data

The field has a value = 1 to the West of the island and a value = 2 to the East (Fig. 4.1). The number of data to the West is four times the number of data to the East.

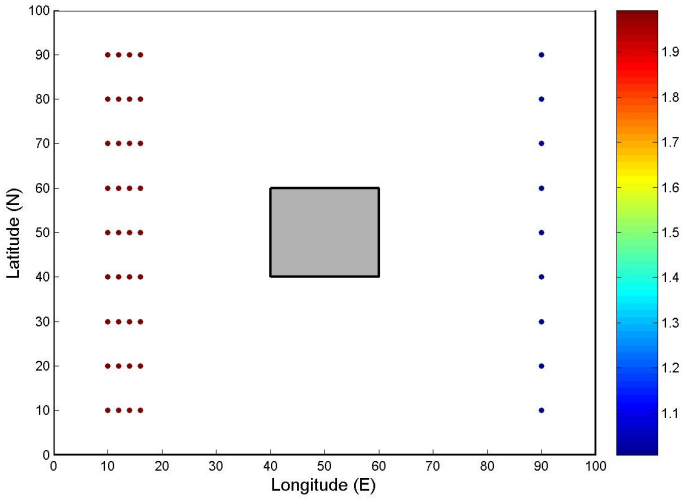


Figure 4.1: Island case data.

4.1.2 Mesh

You can observe that the mesh covers only the region if interest, *i.e.* domain where we have water.

ℓ_m	Surf. coef.	Smooth num.
5	1.5	5

Table 4.1: Island mesh parameters.

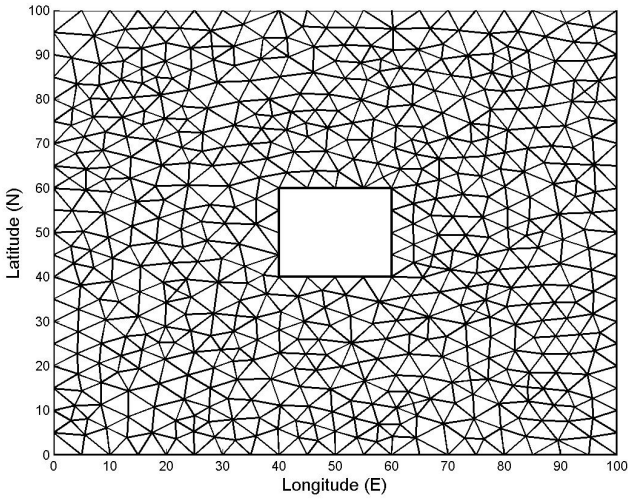


Figure 4.2: Island case mesh.

4.1.3 Analysis

Fig. 4.3 shows that the error is lower where the data coverage is higher, but the analysis itself is not biased.

ℓ_c	σ	Ref. Field	Δx	Δy
15	100	Mean value	1	1

Table 4.2: Island analysis parameters

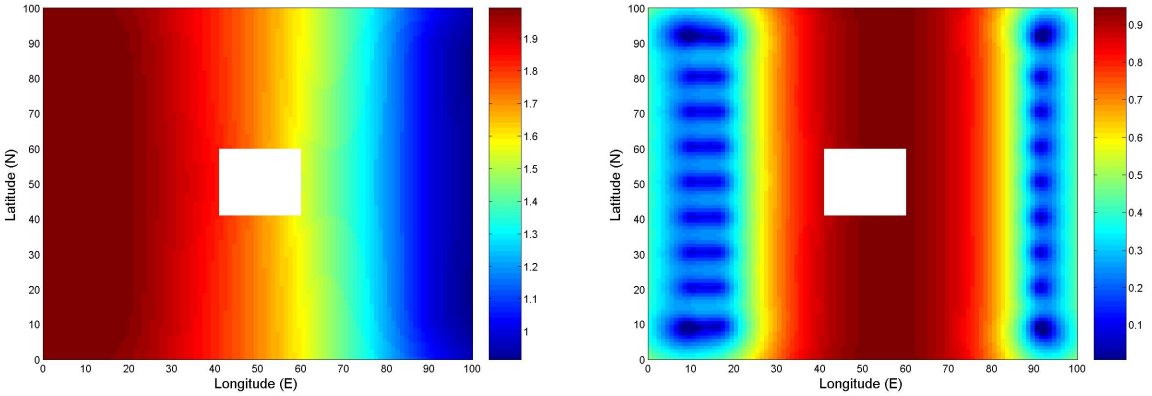


Figure 4.3: Island analysis and error fields.

4.2 Peninsula

In this case, we have a domain that separates the sea in two regions.

4.2.1 Data

There is one data on each side of the peninsula: to the left of the peninsula, the field value is 10 and to the right, the value is 5.

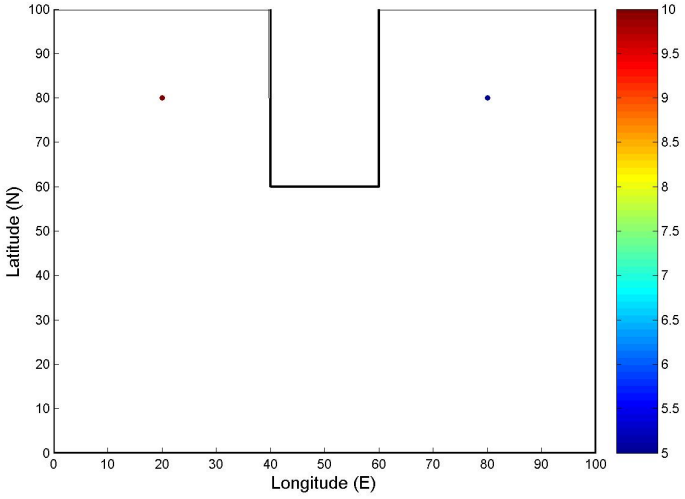


Figure 4.4: Peninsula case data.

4.2.2 Mesh

ℓ_m	Surf. coef.	Smooth num.
5	1.5	3

Table 4.3: Peninsula mesh parameters

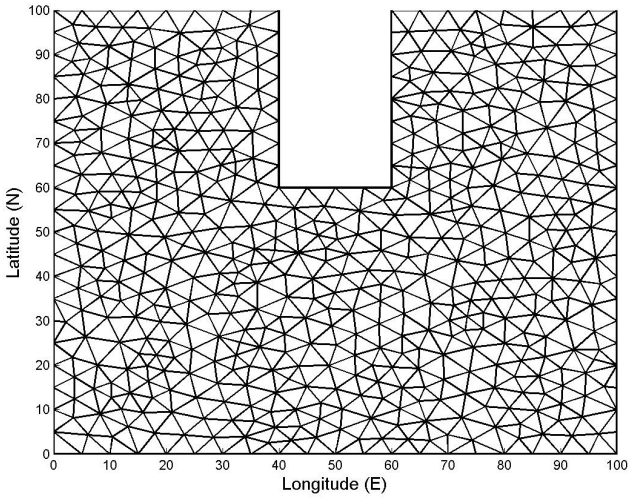


Figure 4.5: Peninsula case mesh.

4.2.3 Analysis

The region of influence of the data is separated by the peninsula. The error field is symmetric.

ℓ_c	σ	Ref. Field	Δx	Δy
15	100	Mean value	1	1

Table 4.4: Peninsula analysis parameters

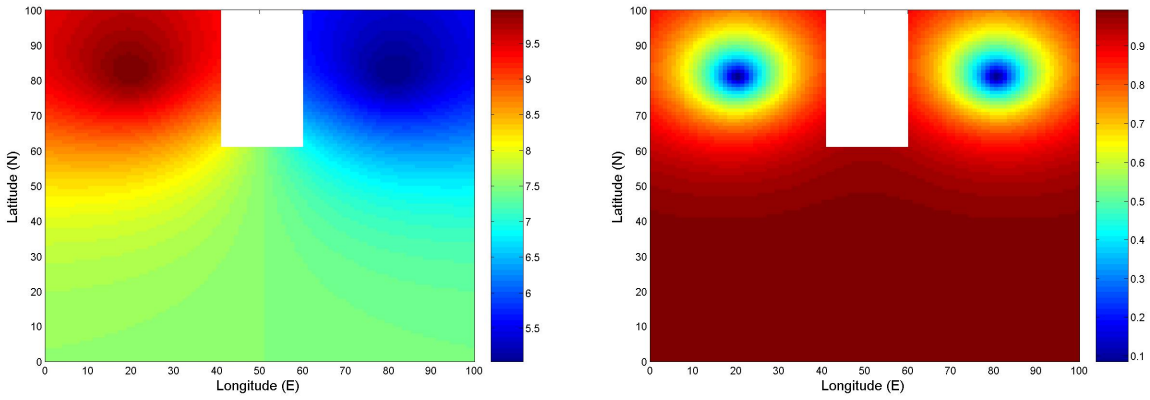


Figure 4.6: Peninsula analysis and error fields.

4.3 Interior sea

Here the two domains are completely separated by a boundary of land.

4.3.1 Data

One data is located inside the interior domain and another one is in the West side of the exterior domain.

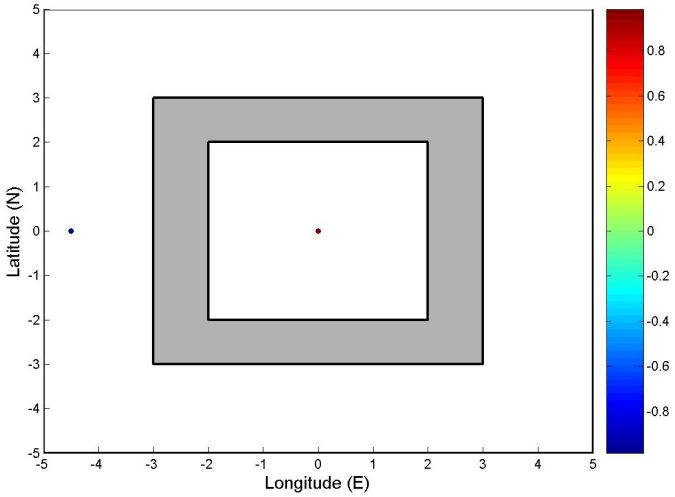


Figure 4.7: Lake case data.

4.3.2 Mesh

The meshes are generated in the two regions.

ℓ_m	Surf. coef.	Smooth num.
0.5	1.5	3

Table 4.5: Lake mesh parameters

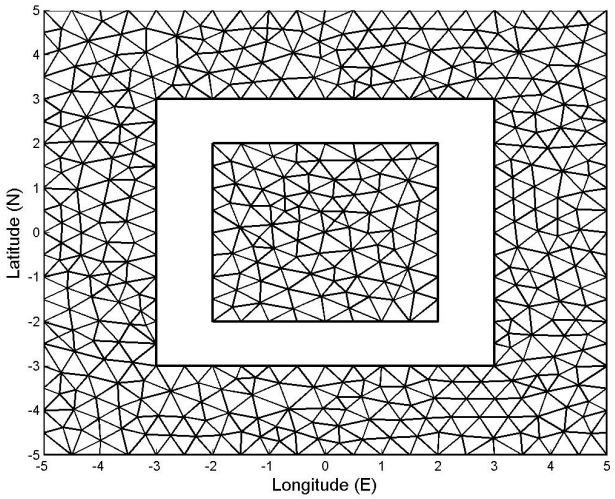


Figure 4.8: Lake case mesh.

4.3.3 Analysis

The analysed fields in the interior and the exterior region are not influenced one by each other

The fields and error fields are disconnected.

ℓ_c	σ	Ref. Field	Δx	Δy
1.5	100	Mean value	0.1	0.1

Table 4.6: Lake analysis parameters

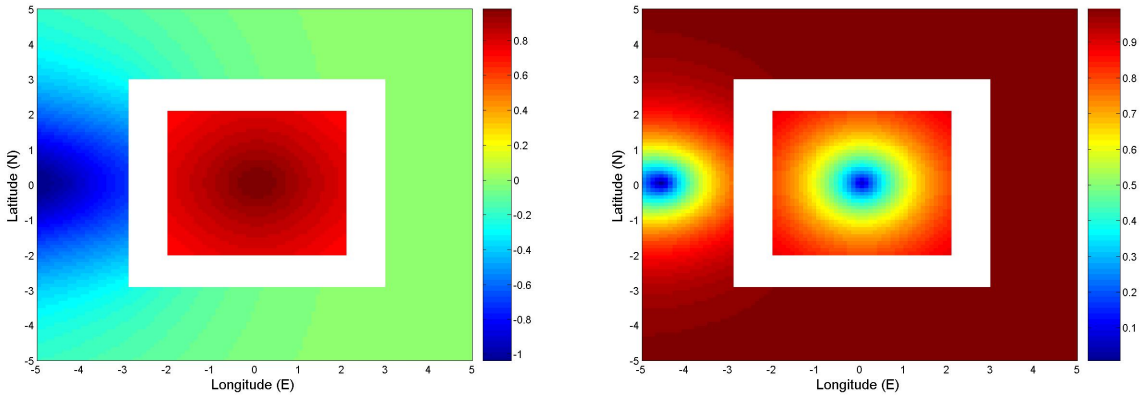


Figure 4.9: Lake analysis and error fields.

4.4 Noise

4.4.1 Data

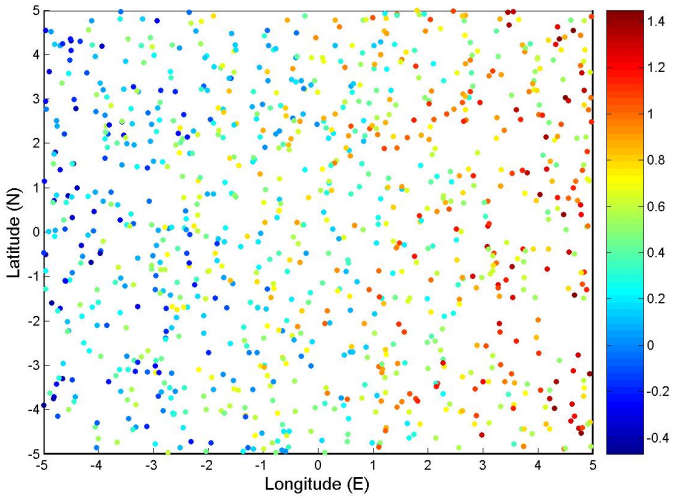


Figure 4.10: Noise case data.

4.4.2 Mesh

ℓ_m	Surf. coef.	Smooth num.
0.5	1.5	3

Table 4.7: Noise mesh parameters

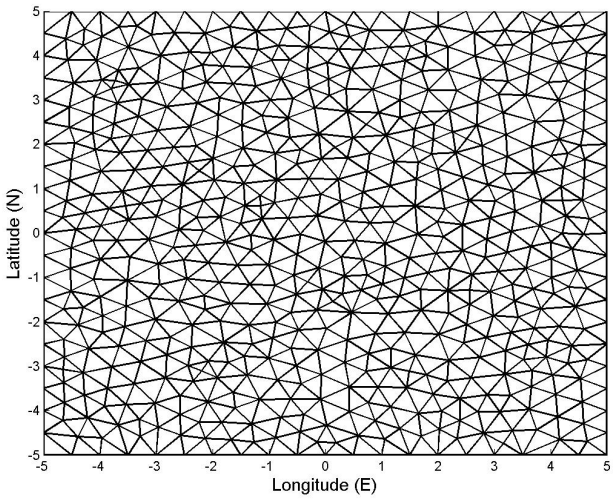


Figure 4.11: Noise case mesh.

4.4.3 Analysis

ℓ_c	σ	Ref. Field	Δx	Δy
1.5	10	Mean value	0.1	0.1

Table 4.8: Noise analysis parameters

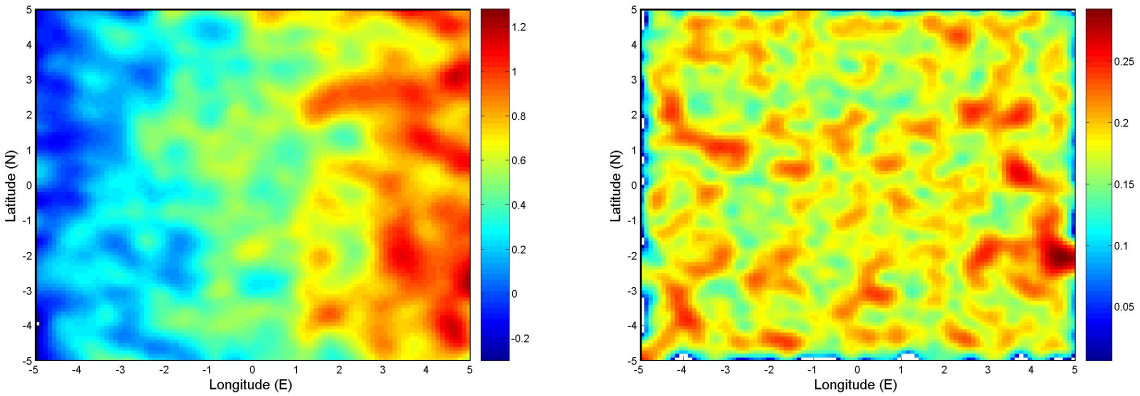


Figure 4.12: Noise analysis and error fields.

4.5 Single point

4.5.1 Data

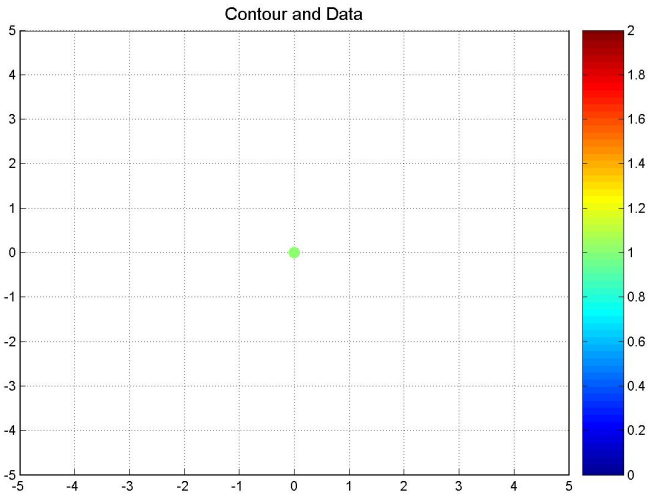


Figure 4.13: Single point case data.

4.5.2 Mesh

ℓ_m	Surf. coef.	Smooth num.
1/3	1.5	5

Table 4.9: Single point mesh parameters

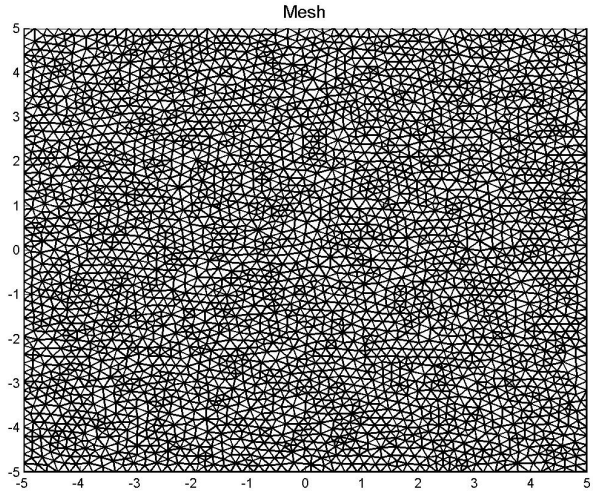


Figure 4.14: Single point case mesh.

4.5.3 Analysis

ℓ_c	σ	Ref. Field	Δx	Δy
1	1000	No	0.2	0.2

Table 4.10: Single point analysis parameters

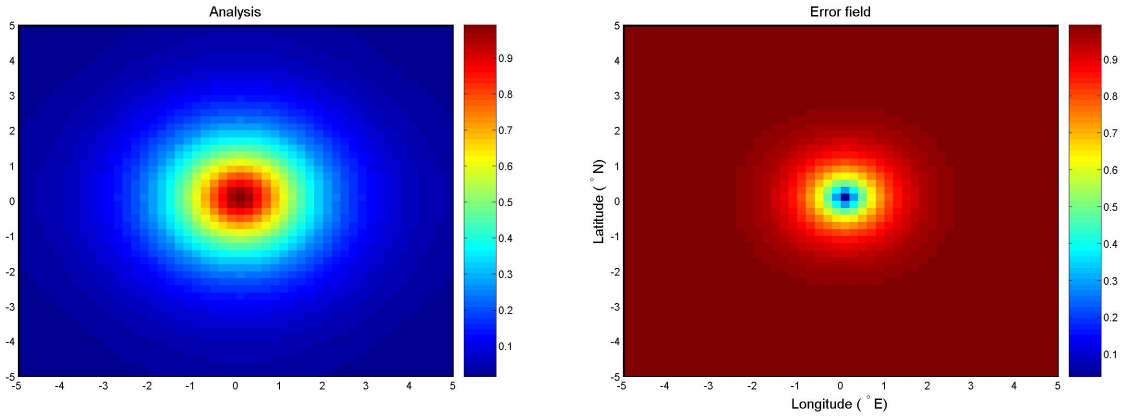


Figure 4.15: Single point analysis and error fields.

5

A complete example with the GUI

This chapter describes the procedure to obtain an analysed field step-by-step using the GUI. Note that the first step (data extraction) is not necessary if you already have a data set at your disposal.

5.1 Data Extraction

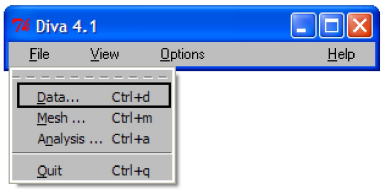


Figure 5.1: Data extraction.

In the Data Extraction window, choose the following files

- Data Base Description File: `liste.des`,
- Generic data output file name: `adriatic.data`,

Then specify the time (*Years* and *Period*) and space domains where you want to perform the extraction. If you use a contour to delimitate the region you want to work in, load `adriatic.cont` as Contour description file.

Finally choose the Data type and nature as well as the quality criterion.

Once you have filled all the fields of the window, press the *Save configuration* button. This will allow you to load automatically the parameters you saved this way, with the *Load configuration* command.

The extraction is performed by clicking on the *OK* button. The data file is displayed with the *Display data file* command. The result obtained with the same parameters as the fig. 3.3 is showed on the fig. 5.2.

To visualize the result, open a Matlab session, go in the directory `diva4.1/src/Matlab/GUI` and execute the script `diva_contour`.

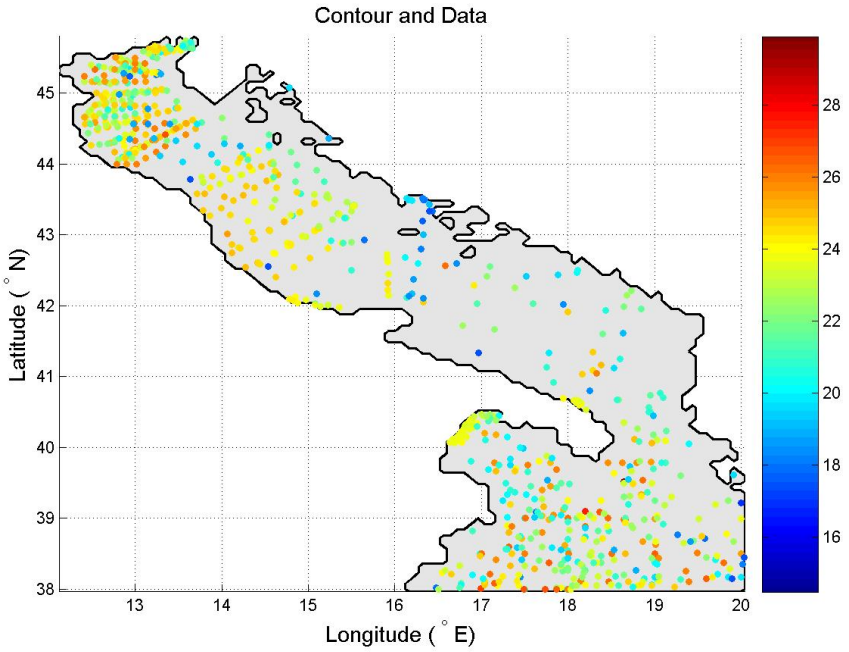


Figure 5.2: Adriatic sea contour and data.

5.2 Mesh generation

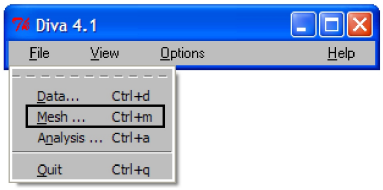


Figure 5.3: Mesh generation.

Select the input and output files for the mesh generation:

- Contour description File: `liste.cont`,
- Mesh file name: `adriatic.mesh`,

Decide the reference length, surface coefficient and the smooth number you want to use. The suggested values for that case are:

Reference length = 0.5
Surface coef. = 1.5
Smooth number = 3

These mesh parameters can be modified in order to see their influence on the mesh shape.

Once the parameters are chosen, use the *OK* button to create the mesh. You can examine the mesh file with the command *Display Mesh File*. The graphic result is showed on fig. 5.4.

For the visualization, use the Matlab script `diva_mesh`.

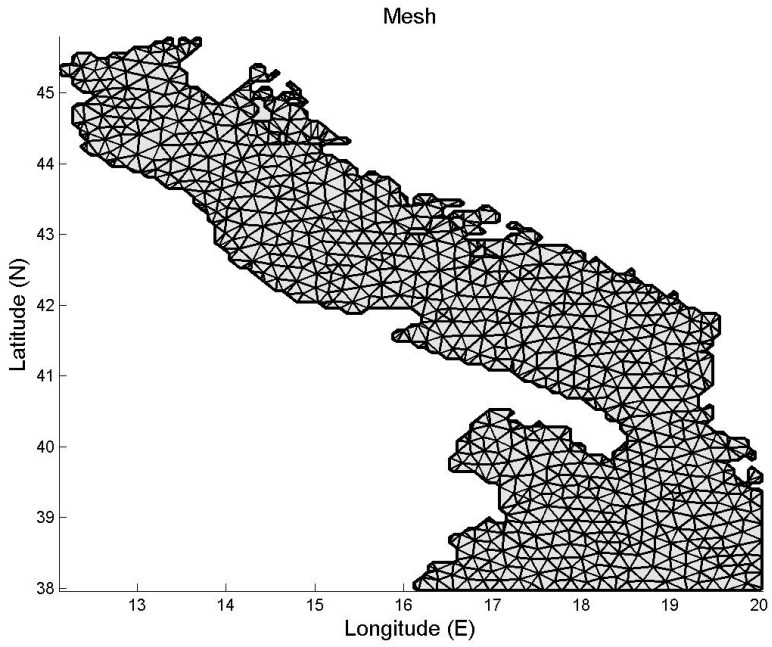


Figure 5.4: Adriatic sea mesh.

5.3 Analysis

In the Analysis window, select the Data file you created with the extraction (`adriatic.dat`) and the mesh you generated (`adriatic.mesh`). Choose an output file name (`adriatic.anl` for example).

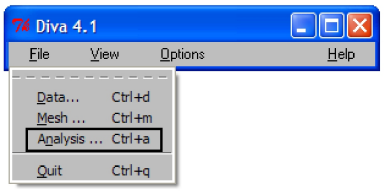


Figure 5.5: Mesh generation.

The analysis parameters can be chosen according to the fig. 3.5.

Save you configuration (`adriatic_analysis.conf`) when all the parameters are set.

The resulting analysed and error fields are showed on figs. 5.6 and 5.7. Observe that the error field is lower where the data density is higher.

To get the graphical output, you can use either the Matlabscripts `diva_analysis.m` and `diva_error.m`, or the NetCDF visualization softwares, as described in section 3.6.

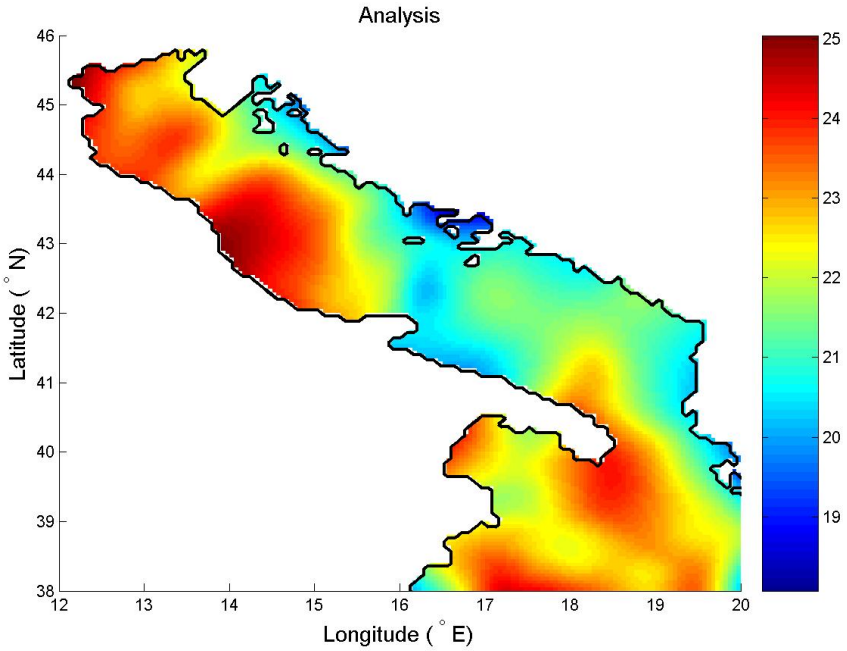


Figure 5.6: Adriatic sea analysed field.

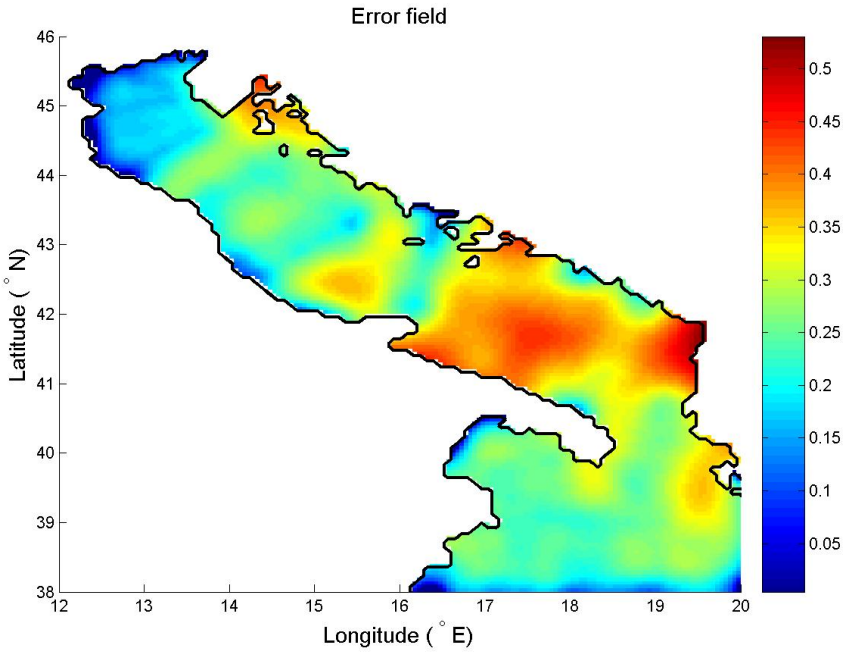


Figure 5.7: Adriatic sea error field.

6

Some properties of Generalized Cross Validation. Application to Diva

6.1 Introduction

The purpose of this chapter is to describe the validations tools included in the Diva software. We start with theorical notions about the Cross Validation, followed by a section about the quality control of data. The chapter ends with the application to Diva.

6.1.1 Generalities

Let us consider the vector $\mathbf{d}$ containing the N data anomalies. Objective analysis of $\mathbf{d}$ leads to analysed field with minimal expected error variance. The analysis ϕ^a at any location $\mathbf{r}$ is given by

$$\phi^a(\mathbf{r}) = \mathbf{c}(\mathbf{B} + \mathbf{R})^{-1}\mathbf{d} \tag{6.1}$$

where $\mathbf{c}$ is a vector containing the background covariance between the point in which the analysis is to be performed and all data point locations. The optimal interpolation is based on the *background covariance* matrix $\mathbf{B}$ and *error covariance* matrix $\mathbf{R}$ of the data.

Let us call $\tilde{\mathbf{d}}$ the analysis vector at data points. The two vectors $\mathbf{d}$ and $\tilde{\mathbf{d}}$ can be related by the expression:

$$\tilde{\mathbf{d}} = \mathbf{A}\mathbf{d} \tag{6.2}$$

where the matrix $\mathbf{A}$, used to perform the analysis at the data points, is calculated according to

$$\mathbf{A} = \mathbf{B}(\mathbf{B} + \mathbf{R})^{-1}. \tag{6.3}$$

The *data-covariance* matrix is the statistical average $\langle \quad \rangle$ of data products:

$$\left\langle \mathbf{d}\,\mathbf{d}^{\mathsf{T}} \right\rangle = \mathbf{B} + \mathbf{R}, \tag{6.4}$$

where $^{\mathsf{T}}$ it the transposed matrix or vector. For uncorrelated observational errors, error-covariance matrix $\mathbf{R}$ is diagonal, with a variance ϵ_i^2 for point i, *i.e.*

$$\mathbf{R} = \mathrm{diag}(\epsilon_i^2)$$

In that case, we can show that the variance of expected misfit at point i is

$$\left\langle \left(d_i - \tilde{d}_i \right)^2 \right\rangle = \epsilon_i^2 (1 - A_{ii}). \tag{6.5}$$

In practice covariance matrices are known only imperfectly: their structure is often considered to be fixed, but with imperfectly known amplitude. In other words it is often assumed that

$$\mathbf{B} = \sigma^2 \hat{\mathbf{B}}, \tag{6.6a}$$
$$\mathbf{R} = \epsilon^2 \hat{\mathbf{R}}, \tag{6.6b}$$
$$\frac{\mathbf{d}^{\mathsf{T}}\mathbf{d}}{N} = \sigma^2 + \epsilon^2, \tag{6.6c}$$

where $\hat{}$ matrices are fixed and non-dimensional, while the field variance σ^2 and the error variance ϵ^2 are imperfectly known.

By definition of the average error- and field-variance, we have:

$$\frac{1}{N}\sum_{i=1}^{N} \epsilon_i^2 = \frac{1}{N}\mathrm{trace}\,(\mathbf{R}) = \epsilon^2 \quad \Rightarrow \frac{1}{N}\mathrm{trace}\left(\hat{\mathbf{R}}\right) = 1 \tag{6.7}$$
$$\frac{1}{N}\sigma_i^2 = \frac{1}{N}\mathrm{trace}\,(\mathbf{B}) = \sigma^2 \quad \Rightarrow \frac{1}{N}\mathrm{trace}\left(\hat{\mathbf{B}}\right) = 1 \tag{6.8}$$

The unknown parameter that controls the analysis is the ratio of the signal and noise variances, called *signal-to-noise ratio*:

$$\lambda = \frac{\sigma^2}{\epsilon^2}, \tag{6.9}$$

because matrix $\mathbf{A}$ depends only on λ:

$$\mathbf{A}(\lambda) = \hat{\mathbf{B}}(\hat{\mathbf{B}} + \lambda^{-1}\hat{\mathbf{R}})^{-1}. \tag{6.10}$$

Dividing both sides of Eq. 6.6c by σ^2 and ϵ^2, we also find that

$$\sigma^2 = \frac{\lambda}{1+\lambda}\frac{\mathbf{d}^\mathsf{T}\mathbf{d}}{N}, \tag{6.11}$$

$$\epsilon^2 = \frac{1}{1+\lambda}\frac{\mathbf{d}^\mathsf{T}\mathbf{d}}{N}, \tag{6.12}$$

so that knowing λ we can calculate the signal and noise variances from the data values.

6.1.2 Ordinary Cross Validation (OCV)

We want to optimize the parameter λ by looking for its value for which the analysis has a minimal error. For this reason, we need to find a proxy norm that we will be able to minimize. As the difference of the analysis with the true field is not available, we will work with the difference of the analysed field at the data points with respect to the original data field:

$$\theta_i^2 = (d_i - \tilde{d}_i)^2. \tag{6.13}$$

If we try to minimize this norm, we will get an infinite signal-to-noise ratio and a perfect data-analysis fit. This is because the analysis at the data point is directly influenced by the corresponding data.

To avoid this inconvenient, the solution is to calculate the difference of the data value with respect to the analysed field in which the data under investigation was not taken into account. This is called the *Ordinary Cross Validation*.

To make this estimate robust, the analysis has to be repeated over a large number of data points, increasing the computing cost, so that OCV is generally too expensive to perform.

6.1.3 Generalized Cross Validation (GCV)

According to [Craven and Wahba (1979)], modifying the error estimate as follows:

$$\hat{\theta}_i^2 = \frac{(d_i - \tilde{d}_i)^2}{(1 - A_{ii})^2}. \tag{6.14}$$

allows to keep the data during the analysis and will reduce the computing cost. In this formulation, the denominator penalizes more heavily data points in which the analysis is forced to be close the data and accounts therefore for the self-influence of the data point (which is absent in the case of pure cross-validation).

Computation of A_{ii}

When the matrix $\mathbf{A}$ is not explicitly calculated, A_{ii} can be obtained by performing an analysis with a vector $\mathbf{e}_i = (0\,0\ldots0\,1\,0\ldots0)$ (zero on all data locations, except at point i, where its value is one). This demands an analysis for every data point in which the estimator is constructed.

The associated computational cost can be reduced by replacing A_{ii} by the average value and assuming:

$$A_{ii} \simeq \frac{1}{N}\text{trace}\,(\mathbf{A})\,. \tag{6.15}$$

To avoid calculating all A_{ii} and summing them up, we can use the following estimate [Girard (1989)]:

$$\frac{1}{N}\text{trace}\,(\mathbf{A}) \simeq \frac{\mathbf{z}^\mathsf{T}\mathbf{A}\mathbf{z}}{\mathbf{z}^\mathsf{T}\mathbf{z}} \tag{6.16}$$

where $\mathbf{z}$ is a vector of random variables of zero mean. For robustness, the trace estimate can be repeated several times with different random vectors, averaging of the different estimates. The number of estimates is the parameter provided to the module `GCVFAC` of Diva.

Even if $\mathbf{A}$ is not available, $\mathbf{A}\mathbf{z}$ can be calculated easily by applying the analysis tool to a random vector and retrieve the analysis of this random vector on the data locations. Then the scalar product of the analysis of the random vector with the original random data provides the numerator of (6.16) while the denominator is simply the squared norm of the random vector.

Generalized cross validator

In order to make the error estimator robust, we take the average over all data points and define the *generalized cross validator* as

$$\Theta^2 = \frac{1}{N}\sum_{i=1}^{N}\hat{\theta}_i^2.$$

Assuming temporarily $\epsilon_i^2 = \epsilon^2$, hence having all misfits with the same weight, the generalized cross validator is obtained:

$$\Theta^2 = \frac{\|\,\mathbf{d} - \tilde{\mathbf{d}}\,\|^2}{N\left(1 - \frac{1}{N}\text{trace}\,(\mathbf{A})\right)^2} = \frac{\|\,(\mathbf{I}-\mathbf{A})\mathbf{d}\,\|^2}{(1/N)\left(\text{trace}\,(\mathbf{I}-\mathbf{A})\right)^2} \tag{6.17}$$

The GCV consists in minimizing Θ^2 by changing the signal-to-noise ratio λ. Θ^2 is a global estimate of the analysis error variance.

In view of (6.5) and (6.17) the expected variance of the noise can also, by assuming a spatial average corresponds to a statistical expectation, be calculated as

$$\epsilon^2 \simeq \Theta^2\left(1 - \frac{1}{N}\text{trace}\,(\mathbf{A})\right) \tag{6.18}$$

When the observational errors are uncorrelated but vary in space, we should replace the residual measure $r = (\mathbf{d} - \tilde{\mathbf{d}})^{\mathsf{T}}(\mathbf{d} - \tilde{\mathbf{d}})$ by

$$r = (\mathbf{d} - \tilde{\mathbf{d}})^{\mathsf{T}} \hat{\mathbf{R}}^{-1} (\mathbf{d} - \tilde{\mathbf{d}}) \tag{6.19}$$

to take into account the relative noise level.

For a diagonal matrix, we can define weights w_i such that

$$\epsilon_i^2 = \frac{\epsilon^2}{w_i} = \frac{\sigma^2}{\lambda}\frac{1}{w_i} \tag{6.20}$$

Defining the diagonal matrix $\mathbf{W} = \mathrm{diag}(w_i)$, the generalized cross validator then reads:

$$\Theta^2 = \frac{\left(\mathbf{d} - \tilde{\mathbf{d}}\right)^{\mathsf{T}} \mathbf{W} \left(\mathbf{d} - \tilde{\mathbf{d}}\right)}{N\left(1 - \frac{1}{N}\mathrm{trace}\,(\mathbf{A})\right)^2} \tag{6.21}$$

where the weights should, according to (6.7) satisfy

$$\sum_i \frac{1}{\tilde{w}_i} = N. \tag{6.22}$$

The $\tilde{w}_i$ are the values provided as optional fourth column in `data.dat`.

6.2 Quality control (QC) of data

Having defined the generalized cross validator, we look for a criterion that will allow us to reject or accept a given data.

6.2.1 Quality criteria

The first possibility is to compare the actual value of the misfit with the expected standard deviation $\left\langle \left(d_i - \tilde{d}_i\right)^2 \right\rangle$, leading to the criterion:

$$|d_i - \tilde{d}_i| > 3\Delta_i^{(1)} \tag{6.23}$$

$$\text{with} \qquad \Delta_i^{(1)} = \epsilon_i \sqrt{1 - A_{ii}}, \tag{6.24}$$

which is the most expensive version if A_{ii} is not explicitly known and must be evaluated by analysis of vectors such as $\mathbf{e}_i = (0\,0\ldots 0\,1\,0\ldots 0)$.

If we replace A_{ii} by its average $\frac{1}{N}\mathrm{trace}\,(\mathbf{A})$, we have a second criterion based on:

$$\Delta_i^{(2)} = \epsilon_i \sqrt{\left(1 - \frac{1}{N}\mathrm{trace}\,(\mathbf{A})\right)}. \tag{6.25}$$

This version requires only a few analysis of a random vector if trace $(\mathbf{A})$ cannot be evaluated explicitly.

Finally, in case the noise is not calculated from Θ^2, another estimate is then, according to (6.18)

$$\Delta_i^{(3)} = \frac{\epsilon_i}{\epsilon}\left(1 - \frac{1}{N}\text{trace}\,(\mathbf{A})\right)\Theta, \tag{6.26}$$

which we can calculate directly from the rms value of the misfit (or residual) and the generalized cross validator Θ. This version can easily be used simultaneously with test (6.25) using the results of the GCV.

6.2.2 Normalized version

Let us consider the scaled variable s defined as

$$s_i = \frac{d_i - \tilde{d}_i}{\Delta_i}. \tag{6.27}$$

Because we will look for outliers, instead of working with mean values and variances, more robust estimators are the median and MAD (*median absolute deviation*).

A bias in the analysis is likely to exist if $S = \text{median}(s_i)$ is not much smaller than one (the typical values of s should be of the order of one if the misfit estimate is correct on average).

Instead of calculating the standard deviation of s we calculate the median absolute deviation δ defined as

$$\delta = \text{MAD}(s_i) = 1.4826\,\text{median}|s_i - S| \tag{6.28}$$

where the factor 1.4826 is introduced such that for a normal distribution, the MAD is identical to the standard deviation.

Hence we can detect points qualified as outliers if they satisfy the inequality

$$|s_i - S| \geq 3\delta. \tag{6.29}$$

This normalized version of quality control to some extent corrects for inaccurate estimation of the expected misfits Δ.

6.3 Application and implementation in Diva

The latest version of Diva allows the user to perform generalized cross validation as well as quality control. Two new modules were added to achieve these tasks: `dataqc` and `gcvfac`.

6.3.1 Diva formulation

The Diva theory is extensively described in Chapter 1. We limit ourselves here to recall the base formulation.

When working with non-dimensional space coordinates, Diva consists in minimizing

$$\begin{aligned} J[\varphi] &= \sum_{i=1}^{N} \mu_i L^2 [d_i - \varphi(x_i, y_i)]^2 \\ &+ \int_{\tilde{D}} \left(\tilde{\nabla}\tilde{\nabla}\varphi : \tilde{\nabla}\tilde{\nabla}\varphi + \alpha_1 L^2 \tilde{\nabla}\varphi \cdot \tilde{\nabla}\varphi + \alpha_0 L^4 \varphi^2 \right) \mathrm{d}\tilde{D} \end{aligned} \tag{6.30}$$

where $\varphi(x_i, y_i)$ is the field to analyse, d_i is the field value at point (x_i, y_i), L is the characteristic length of the problem and α_0, α_1 and μ are parameters that have to be adapted to the considered problem.

α_0 fixes the length scale L over which variations are significant to move the kernel function from one to zero:

$$\alpha_0 L^4 = 1 \tag{6.31}$$

μL^2 fixes the weight on data vs. regularization. As the signal variance is σ^2 and the noise variance ϵ_i^2, we can write:

$$\mu_i L^2 = 4\pi \frac{\sigma^2}{\epsilon_i^2} \tag{6.32}$$

α_1 fixes the influence of gradients:

$$\alpha_1 L^2 = 2\xi \tag{6.33}$$

where ξ is a non-dimensional parameter close to one if the gradients are to be penalized with a similar weight than the second derivatives.

6.3.2 Implementation of result validation in Diva

The overall signal-to-noise ratio is calculated as

$$\lambda = \frac{\sigma^2}{\epsilon^2}. \tag{6.34}$$

The relative weights on the data are then defined as

$$\tilde{w}_i = \frac{\mu_i L^2}{4\pi\lambda}. \tag{6.35}$$

To be coherent with the total noise we should in principle use relative weights [1] on the data that satisfy (6.22).

Hence, defining $\bar{w}$ as

$$\frac{1}{\bar{w}} = \frac{1}{N}\sum_i \frac{1}{\tilde{w}_i}, \tag{6.36}$$

the weights $w_i = \tilde{w}_i/\bar{w}$ are relative weights coherent with the noise.

In this case, we can use (6.21) for the estimator.

In practice, in Diva, the relative weights can be retrieved by $w_i = \mu_i/\bar{\mu}$, where the average $^-$ is a harmonic mean. The minimisation and the optimum for λ are the same, even if the weights provided by the user do not satisfy the condition (6.22). However the interpretation of Θ, ϵ and σ might be different.

Diva GCV

In Diva, the matrix $\mathbf{A}$ is not explicitly constructed, but the product $\mathbf{Ax}$ consists simply in the application of the analysis in data locations.

The value of trace$(\mathbf{A})$ can be evaluated and stored for subsequent quality control by (6.25) and (6.26). For quality control (6.24), A_{ii} must be evaluated by an analysis of the pseudo-data vector $\mathbf{e}_i$ (zeros everywhere except unit value in point i). The analysis can benefit from the LU decomposition already performed for previous analysis.

Diva QC

Quality control with Diva can be performed according to one of the three criteria (6.24), (6.25) or (6.26), respectively implemented in Diva with `divaqc`, `divaqcbis` and `divaqcter`. The corresponding outputs are given in `outliers.dat`, `outliersbis.dat` and `outlierster.dat`).

The modules `divaqc*` also generate `outliers*.normalized.dat`, which contain, in a sorted way (from the most suspect data to less suspect), the possible outliers from the normalized misfits test (6.29).

By default, the criterion used in `divadress` is `divaqcbis`, but you can change it by editing the file `divadress` and replacing `divaqcbis` by one of the other quality test (`divaqc` or `divaqcter`).

[1] The optional fourth column for data in Diva

Part II

Appendix

A

Diva code

THE engine of Diva is a set of Fortran 77 subroutines driven by local input `fort.*` files and local output `fort.*` files.

The GUI is driven by programs written in Tcl/Tk (see Sec. A.3).

This chapter is dedicated to the description of these programs, in order to allow the advanced user to adapt the code.

A.1 Fortran code

The Fortran programs are sorted according to their purpose: Analysis, Mesh Generation, Graphical Interface, NetCDF and Utilities.

A.1.1 Diva programs [src/Fortran/Calc]

The main file of the Fortran code is `diva.f`: this routine calls successively the subroutines enumerated below, in order to perform a Diva analysis using the finite element method:

`mathpr.f` : describes the mathematical problem.

`topolo.f` : describes the topology of the finite element grid.

`meshgn.f` : generates a square finite element mesh on a regular grid.

`datapr.f` : input of data to be fitted by spline smoothing.

`bcondi.f` : Dirichlet boundary conditions to be fixed.

`constr.f` : input of information for constraint implementation.

`solver.f` : builds and solves the global linear finite element system.

`stores.f` : storage of the solution.

`esterr.f` : estimates the analysis error (same grid as the analysis).

`coord.f` : coordinate change (longitude, latitude) to (x, y) and (x, y) to (longitude, latitude) if requested.

`gcvfac.f` : estimates the analysis error by generalized cross-validation.

`dataqc.f` : data quality check: estimates of expected data-analysis differences.

A.1.2 Mesh programs [src/Fortran/Mesh]

To generate the coastline, `contourgen.f` can be used.
For the mesh generation, 2 subroutines are used:
`generopt.f`, which generates a multi-connex 2D mesh with the *Delaunay triangulation*,
`contourcheck.f`, which eliminates identical points and checks the contour format.

A.1.3 GUI programs [src/Fortran/Extensions]

The graphical interface requires some particular routines:

`extract.f` : interface to select or extract Data from MODB and MED formatted Data Bases.

`fem3d.f` is based upon a mesh file and a bathymetry file (regular grid in the GHER format); it creates a result file containing information to determinate if a mesh is in land or is sea, for a given depth.

`concat.f` is used to create the 3D file from the 2D files.

`stiff.f` generates the rigidity file (`fort.60`) in the 3D case.

`mask.f` masked the solution according to the bathymetry and the depth.

`sum.f` writes the sum of two fields (with same dimensions) in GHER format.

`substref.f` subtracts the reference field to the computed solution (only available when using the *semi-normed* reference field).

`header2.f` and `visu.f` are subroutines to visualize the data, the mesh and the solution (requires the PlPlot library). Two versions of `visu.f` are provided: one is located in `diva-4.1/src/Fortran/PlPlot` and the other in `diva-4.1/src/-Fortran/NolPlot`. The first can be compiled only if you have installed the PlPlot library on your computer (for now, only under Linux).

A.1.4 Utilities [`src/Fortran/Util`]

`alpha.f` computes the values of α_0 and α_1.

`calcmu.f` computes the coefficient μ knowing the characteristic length L and the signal-to-noise ratio λ.

`calcest.f` : from relative weight on data points (data file `fort.44`) and generalized cross validator value (available in `./output/gcvval.dat`), produces the file containing the expected misfit value at data locations (`fort.76`), to be used by `lookfor-outliers.a` for outlier detection. Used in `divaqcter`.

`calcestbis.f` : from relative weight on data points (data file `fort.44`) and the trace of the analysis matrix (available in `./output/gcvval.dat`) produces the file containing the expected misfit value at data locations (`fort.76`), to be used by `lookfor-outliers.a` for outlier detection. Used in `divaqcbis`.

`dbdb2diva.f` : makes easier the extraction from a website.

`findmin.f` : from a list of values (SNR, GCV, data-variance) found in `fort.11`, tries to find by local parabolic interpolation the minimum of GCV and the place SNR where it is found.

Points do not need to be ordered. The output (`fort.12`) contains the expected value of SNR in which GCV is minimal as well as the `VARBAK` value. Used in `divagcv`.

`fitlsn.f` : from the data file (`fort.10`) and information on coordinate change and reference field (`fort.11`), tries to fit the *Bessel covariance function* to the data covariance.

Output (`fort.66`) contains the correlation length and an rought estimate of the SNR. The data-covariance over all distances is written out in `fort.99`.

The data covariance as a function of data-distance as well as the corresponding fit over distances up to the correlation length are found in `fort.98`. Used in `divafit`.

`griddef.f` creates the file `GridInfo.dat`, of which the content is used for writing the NetCDF output.

`lceleme.f` computes the mesh characteristic length.

`lookforoutliers.f` checks if there is outliers in the data provided for the analysis.

`topoprep.f` makes easier the generation of the topography using a collection of local measurements.

Remark: grid definition for users is based on an origin which is the first grid point. In the code, `xori` and `yori` are defined by $x_i =$ `xori` $+i\Delta x$. (and is thus shiftet one grid space to the left compared to the user origin). Input file `fort.13` is modified from the `param.par` grid information by `griddef.a` accordingly, while `gridInfo.dat` contains the user grid as defined by `param.par`.

A.1.5 NetCDF output [src/Fortran/NC]

Three files allow to get an output under the NetCDF format [1] : `netcdfoutputfield.f`, `netcdfoutputerror.f`, `netcdfoutput.f`.

The first two routines write the analysed and the error fields in two different NetCDF files: `anlysed_field.nc` and `error_field.nc`. The last programs generates `results.nc` which contains both the analysis and the error fields. The information required for the coordinates (xorigin, y origin, dx, dy, xend, yend) are read from `GridInfo.dat`.

```
1
1
.5
.5
100
100
```

Example file A.1.1: *GridInfo.dat*

A.2 Input and output files

The Fortran executables (files ending by `.a`) work with input and output `fort.*` files:

`generopt.a`

- reads as inputs:
  `fort.10`: coast file, see exemple file with description, case of a square island in a square sea.
  `fort.11`: the parameters required for the mesh generation.
- produces as outputs:
  `fort.22`: finite element mesh, to be copied as `fort.11` for the work of `diva.a`.
  `fort.23`: topological parameters.

[1] Do not forget you need the NetCDF library if you want to compile these files. The library needed by your system can be downloaded from
`http://www.unidata.ucar.edu/software/netcdf/`

```
2
4
0 0
100 0
100 100
0 100
4
40 40
40 60
60 60
60 40
```

Example file A.2.1: *fort.10*

```
0
10
1.5
5
```

Example file A.2.2: *fort.11*

```
1 0. 0.
547 10. 0.
40 20. 0.
...
161
49
139
...
92 -398 90 -583 157 -557
93 -455 95 -584 158 -293
95 -447 78 -562 158 -584
```

Example file A.2.3: *fort.22*

```
158
426
268
```

Example file A.2.4: *fort.23*

`diva.a`

- reads as inputs:
  `fort.10`: divawork organiser, defining which modules to use and how, including several parameters.
  `fort.11`: finite element mesh, copy of the `fort.22` produced by `generopt.a`.
  `fort.12`: α_0 and α_1, calculated as:

$$\alpha_0 = \frac{1}{L^4} \quad \text{and} \quad \alpha_1 = \frac{2}{L^2}.$$

`fort.13`: caracteristics of regular output grid.
`fort.20`: data file; 4 columns: $|X|Y|data(X,Y)|\mu|$, where

$$\mu = 4\,\pi\,\frac{\lambda}{L^2},$$

with λ, the signal-to-noise ratio.
`fort.79`: coordinates where analysed field values are requested; 2 columns separated by space: $|X|Y|$.
`fort.15`: parameter `varbak`, variance of the background field, set as $= 0$ will avoid calculating error field.

```
coord 1
0
mathpr 1
2 -> ITYP
0 -> ISYM
2 -> IPB
topolo 1
158 -> Number of Vertex Nodes
426 -> Number of Interface Nodes
268 -> Number of Elements
datapr 1
1
solver 1
0
stores 1
1 -> 1 if normal or seminormed final, otherwise 3
esterr 1
stopex
```

Example file A.2.5: *fort.10*

```
0.0001 -> alpha0
0.02 -> alpha1
```

Example file A.2.6: *fort.12*

```
0 0 -> xorigin, yorigin
1 1 -> dx, dy
100 100 -> nx, ny
-99999 -> Exclusion Value
```

Example file A.2.7: *fort.13*

```
90 90 1 12.5663706
90 80 1 12.5663706
90 70 1 12.5663706

                                        ...


16 30 2 12.5663706
16 20 2 12.5663706
16 10 2 12.5663706
```

Example file A.2.8: *fort.20*

- produces as outputs:
  `fort.71`: field value at data points given in `fort.20`, ascii format.
  `fort.72`: error value at data points given in `fort.20`, ascii format.
  `fort.73`: error at points given in `fort.79`, ascii format.
  `fort.82`: field value at points given in `fort.79`, ascii format.
  `fort.83`: field on regular grid, ascii format.
  `fort.84`: field on regular grid, gher format.
  `fort.86`: error field on regular grid, ascii format.
  `fort.87`: error field on regular grid, gher format.

A.3 Tcl/Tk code [src/Tcl]

The Tcl/Tk files generate the graphical interface. The main file is `main.tcl`: it generates the main window, sets the paths to the `Tcl`, `GUIwork` and `bin` directories.

The different windows of the interface are created with the help of the following programs:

`menu.tcl` : generates the menu bar for the Diva main menu, with its sub-menus: File, View, Options and Help;

`data.tcl` : builds the "Data" window (Diva/File/Data), which permits the data extraction;

`fem.tcl` : generates the finite elements mesh;

`diva.tcl` : performs the analysis on the data set;

`view.tcl` : defines the procedures for the visualization (menu Diva/View);

`option.tcl` : provides the sub-menu "Options" of the Diva main menu (Diva/Options);

`help.tcl` : builds the "Help" window in the Diva main menu (Diva/Help) and provides useful information about the meaning of the parameters choice and the interface.

The program `varlist.tcl` initializes the data in the dialog boxes and the default access paths; `dictionary.tcl` is a dictionary in English and French; `tablist.tcl`: defines tables used in other subroutines.

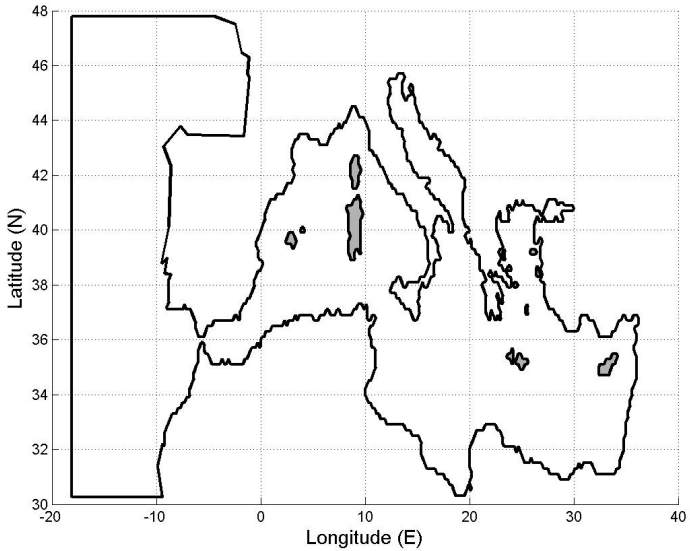
`entry.tcl`, `misc.tcl` and `xmisc.tcl` define several procedures called in other `.tcl` files.

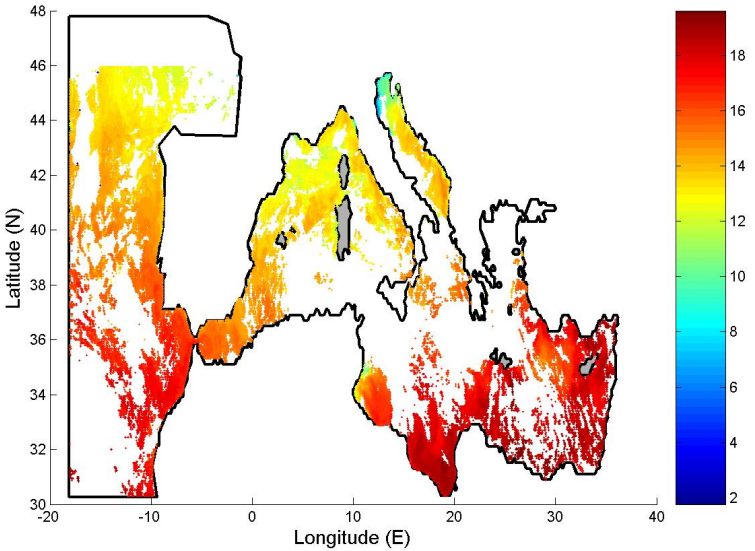
B

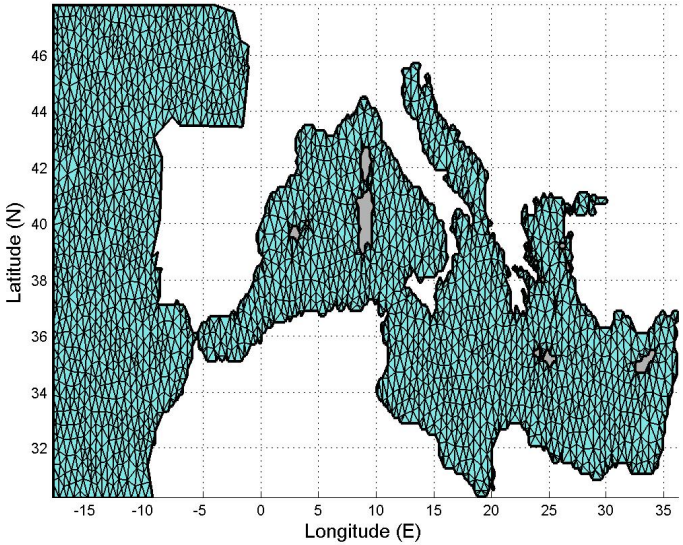
Gallery

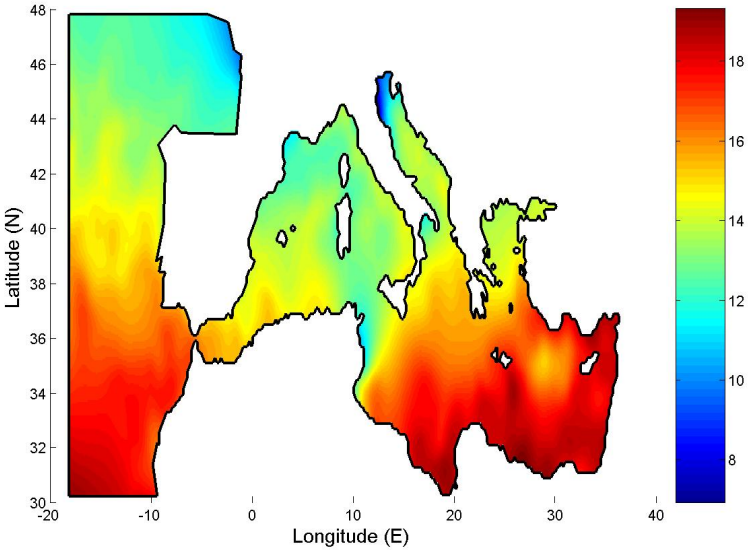
The following results were created with the data of the Diva Workshop participants (*Liège, 13-15 of November, 2006*).

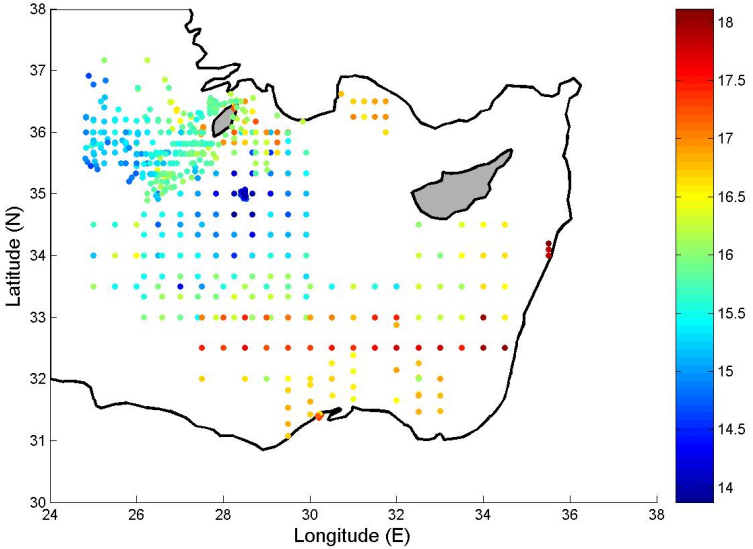
1. Atlantic Ocean and Mediterrranean Sea surface temperature,
2. Eastern Mediterranean temperature,
3. Barents Sea temperature.

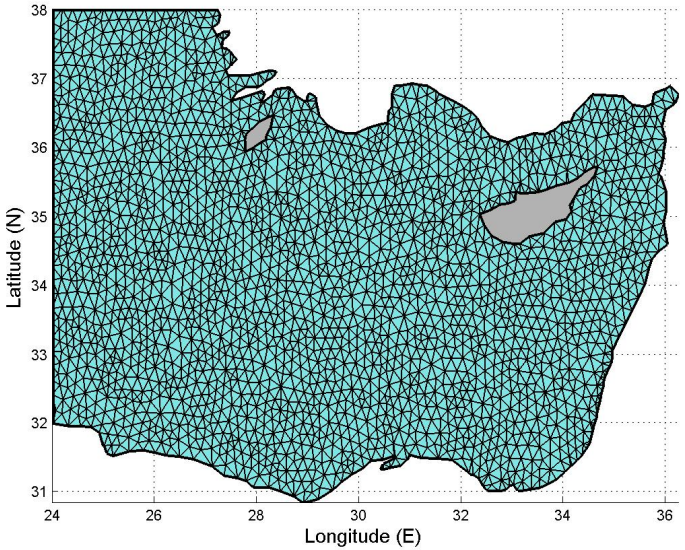

Latitude (N)
Longitude (E)
48
46
44
42
40
38
36
34
32
30
-20
-10
0
10
20
30
40

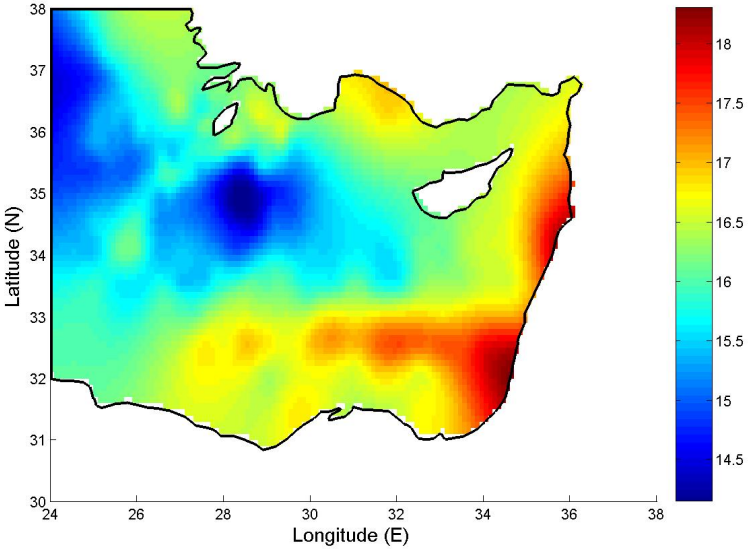

Latitude (N)
Longitude (E)
48
46
44
42
40
38
36
34
32
30
-20
-10
0
10
20
30
40
18
16
14
12
10
8
6
4
2

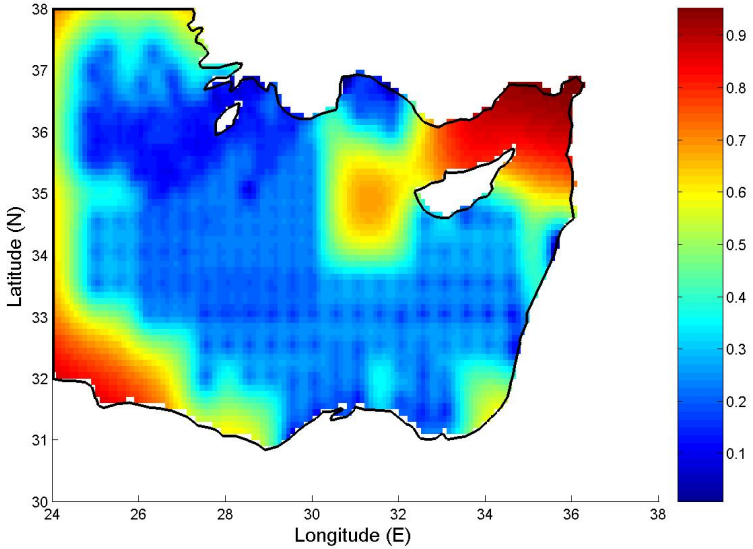

Latitude (N)
Longitude (E)
46
44
42
40
38
36
34
32
-15
-10
-5
0
5
10
15
20
25
30
35

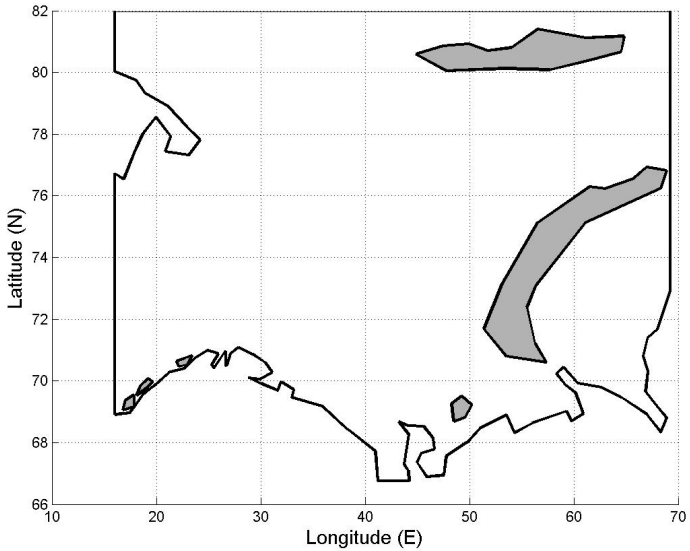

Latitude (N)
Longitude (E)
48
46
44
42
40
38
36
34
32
30
-20
-10
0
10
20
30
40
18
16
14
12
10
8

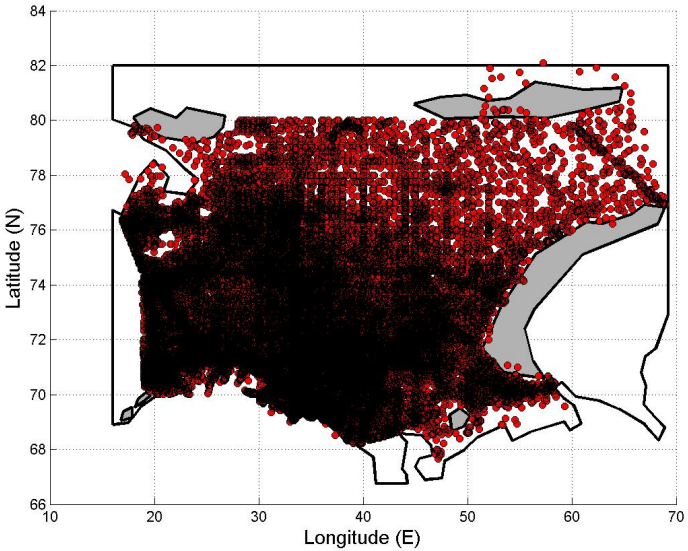

Latitude (N)
Longitude (E)
38
37
36
35
34
33
32
31
30
24
26
28
30
32
34
36
38
18
17.5
17
16.5
16
15.5
15
14.5
14

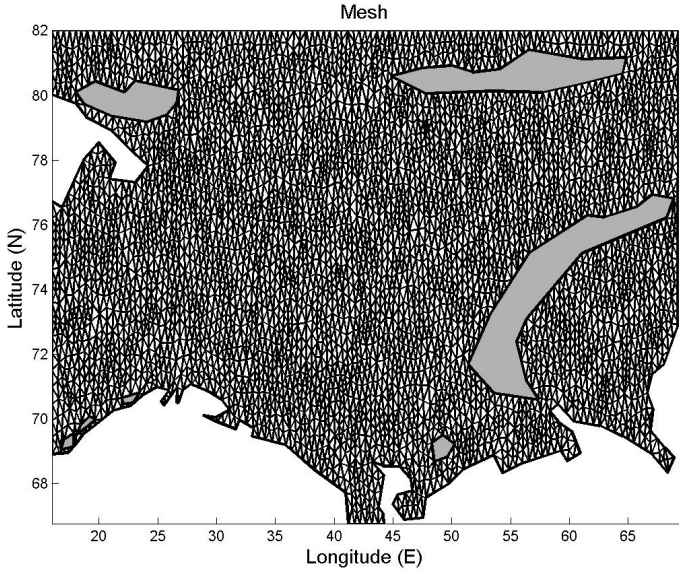

Latitude (N)
Longitude (E)
38
37
36
35
34
33
32
31
24
26
28
30
32
34
36

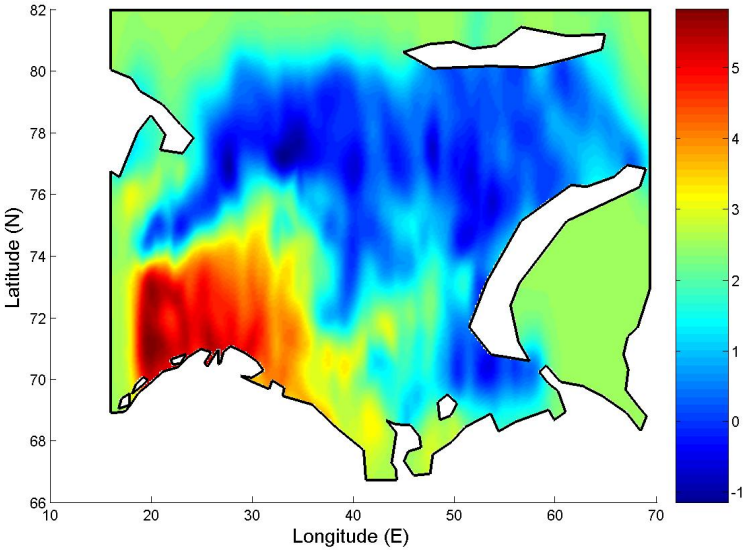

Latitude (N)
Longitude (E)
38
37
36
35
34
33
32
31
30
24
26
28
30
32
34
36
38
18
17.5
17
16.5
16
15.5
15
14.5


Latitude (N)
Longitude (E)
38
37
36
35
34
33
32
31
30
24
26
28
30
32
34
36
38
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1


Latitude (N)
Longitude (E)


Latitude (N)
Longitude (E)


Mesh
Latitude (N)
Longitude (E)


Latitude (N)
Longitude (E)

Bibliography

Diva

[Brankart and Brasseur (1996)] Brankart, J.-M., and P. Brasseur. Optimal Anlysis of In Situ Data in the Western Mediterranean Using Statistics and Cross-Validation. *Journal of Atmospheric and Oceanic Technology*, **13**:477-491, 1996.

[Brankart and Brasseur (1998)] Brankart, J.-M., and P. Brasseur. The general circulation in the Mediterranean Sea: a climatological approach. *Journal of Marine Systems*, **18**:41-70, 1998.

[Brasseur et al. (1996)] Brasseur, P., J.-M. Beckers, J.-M. Brankart, and R. Schoenauen. Seasonal temperature and salinity fields in the Mediterranean Sea: Climatological analyses of a historical data set. *Deep Sea Research*, **43**:159-192, 1996.

[Karafistan et al. (2002)] Karafistan, A., J.-M. Martin, M. Rixen, and J.-M. Beckers. Space and time distributions of phosphates in the Mediterranean Sea. *Deep Sea Research I*, **49**:67-82, 2002.

[Rixen et al. (2000)] Rixen, M., J.-M. Beckers, J.-M. Brankart, and P. Brasseur. A numerically efficient data analysis method with error map generation. *Ocean Modelling*, **2**:45-60,

[Rixen et al. (2001)] Rixen, M., J.-M. Beckers, and J.T. Allen. Diagnosis of vertical velocities with the QG Omega equation: a relocation method to obtain pseudo-synoptic data sets. *Deep Sea Research I*, **48**:1347-1373, 2001.

[Rixen et al. (2005)] Rixen, M., J.-M. Beckers, S. Levitus, J. Antonov, T. Boyer, C. Maillard, M. Fichaut, E. Balopoulos, S. Iona, H. Dooley, M.-J. Garcia, B. Manca, A. Giorgetti, G. Manzella, N. Mikhailov, N. Pinardi, M. Zavatarelli, and the Medar Consortium. The Western Mediterranean Deep Water: a proxy for global climate change. *Geophysical Research Letters*, **32**, 2005.

DINEOF

[Alvera-Azcárate et al. (2005)] Alvera-Azcárate, A., A. Barth, M. Rixen, and J.-M. Beckers. Reconstruction of incomplete oceanographic data sets using Empirical Orthogonal Functions. Application to the Adriatic Sea. *Ocean Modelling*, **9**:325-346, 2005.

[Alvera-Azcárate et al. (2006)] Alvera-Azcárate, A., A. Barth, J.-M. Beckers, and R. Weisberg. Multivariate reconstruction of missing data in sea surface temperature, chlorophyll and wind satellite fields. *Journal of Geophysical Research*, accepted, 2006.

[Beckers and Rixen (2003)] Beckers, J.-M., and M. Rixen. EOF Calculation and Data Filling from Incomplete Oceanographic Datasets. *Journal of Atmospheric and Oceanic Technology*, **20**: 1839-1856, 2003.

[Beckers et al. (2006)] Beckers, J.-M., A. Barth, and A. Alvera-Azcárate. DINEOF reconstruction of clouded images including error maps. Application to the sea-surface temperature around Corsican island. *Ocean Sciences*, **2**:183-199, 2006.

Miscellaneous

[Craven and Wahba (1979)] Craven, P. and G. Wahba. Smoothing noisy data with spline functions. *Numerische Mathematik*, **31**: 377-403, 1979.

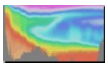
[Girard (1989)] Girard, D. A fast Monte-Carlo cross-validation procedure for large least squares problems with noisy data. *Numerische Mathematik*, **56**: 1-23, 1989.

Useful links

SeaDataNet Home Page
http://www.seadatanet.org/

Diva Home Page
http://modb.oce.ulg.ac.be/modb/diva.html

GHER Home Page
http://modb.oce.ulg.ac.be/



Ocean Data View
http://odv.awi.de/

NetCDF
http://www.unidata.ucar.edu/software/netcdf/

Tcl/Tk
http://www.tcl.tk/

PlPlot
http://plplot.sourceforge.net/

Msys
http://www.mingw.org/msys.shtml

Cygwin
http://www.cygwin.com/

Matlab
http://www.mathworks.com/

University of Liège
http://www.ulg.ac.be/

Conditions of use

Diva is a public domain graphical software and can be found on the GHER 'www' server. This program is subject to conditions of use that can be found on the 'www' server.
Diva is a Liege University software, which will be further developed for SeaDataNet scientific data products in JRA4 activities.
All SeaDataNet products (data and softwares) are freely distributed to the scientific community at the following conditions:

- ☑ The products should be used for scientific purposes only.
- ☑ Articles, papers, or written scientific works of any form, based in whole or in part on data or software supplied by SeaDataNet, will contain a suitable acknowledgement to the SeaDataNet programme of the European Union. The related publications (see bibliography) should also be cited.
- ☑ The applications of the SeaDataNet products are on the full responsibility of the users; neither the Commission of the European Communities nor the SeaDataNet partners shall be held responsible for any consequence resulting from the use of SeaDataNet products.
- ☑ The recipient of these data will accept responsibility of informing all data users of these conditions.
- ☑ The Diva software cannot be incorporated into another software package without permission.

Acknowledgements

This report was written for helping oceanographers to work with the Diva *software. This would not have been possible without the help of scientists involved in Data Analysis projects.*

We would like to thank:

the participant to the Diva *workshop in Liège (November 2006) for their valuable comments to improve the software;*

the GHER *team for its constant support;*

the Fonds de la Recherche Scientifique - FNRS and the Fonds pour la formation à la Recherche dans l'Industrie et dans l'Agriculture (FRIA) for their funding;

the European Union for funding the SeaDataNet project.